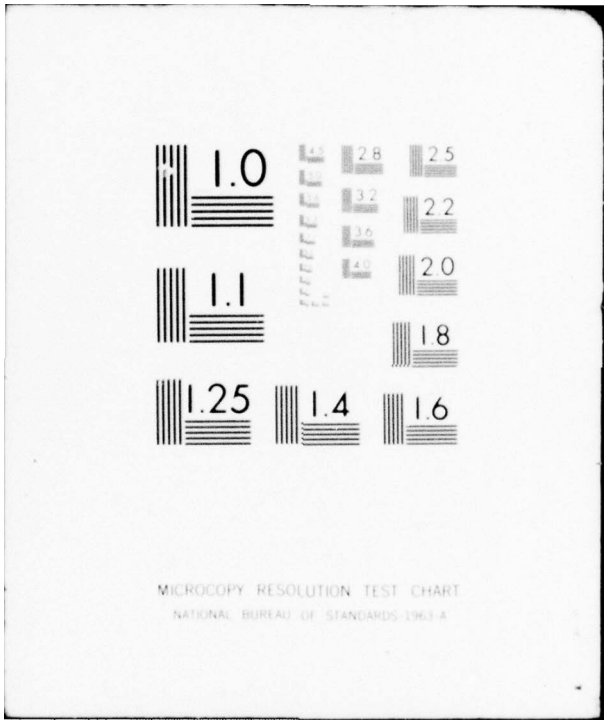
AD-A048 192 AIR FORCE AVIONICS LAB WRIGHT-PATTERSON AFB OHIO F/G 5/3
SOFTWARE COST ESTIMATING METHODOLOGY.(U)
AUG 77 T G JAMES
UNCLASSIFIED AFAL-TR-77-66 NL

1 OF 1
AD A048192

END
DATE FILMED
1-78
DDC


1.0
1.1
1.25
1.4
1.6
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.8
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

AFAL-TR-77-66

SOFTWARE COST ESTIMATING METHODOLOGY

Synthesis and Analysis Branch
System Avionics Division

August 1977

TECHNICAL REPORT AFAL-TR-77-66

Final Report for Period June 1976 - September 1976

Approved for public release; distribution unlimited.

AIR FORCE AVIONICS LABORATORY
AIR FORCE WRIGHT AERONAUTICAL LABORATORIES
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433

FOREWORD

This report covers work conducted in-house by the System Evaluation Group (AAA-3), Synthesis and Analysis Branch, System Avionics Division, Air Force Avionics Laboratory, Wright-Patterson AFB, Ohio 45433, under PE 62204F, Project 2003 "Avionics System Design Technology", Task 200309, "Avionics System Cost-Effectiveness", Work Unit 20030902 "Avionics Life Cycle Cost". The time period of work was June 76 through August 76.

Significant contributors to this report were Capt Boundon and Capt White both from Electronic Systems Division, Hanscom AFB, MA. Great appreciation is extended to Capt. Ken Almquist (AFAL/AAA-3) for his participation.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: AFAL-TR-77-66

2. GOVT ACCESSION NO.:

3. RECIPIENT'S CATALOG NUMBER:

4. TITLE (and Subtitle): SOFTWARE COST ESTIMATING METHODOLOGY

5. TYPE OF REPORT & PERIOD COVERED: Final - June 76 - Sept 76

6. PERFORMING ORG. REPORT NUMBER:

7. AUTHOR(s): Lt Thomas G. James, Jr.

8. CONTRACT OR GRANT NUMBER(s):

9. PERFORMING ORGANIZATION NAME AND ADDRESS:
System Evaluation Group (AFAL/AAA-3)
Air Force Avionics Laboratory
Wright-Patterson AFB, Ohio

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 62204F/2003/09/02

11. CONTROLLING OFFICE NAME AND ADDRESS:
Air Force Avionics Laboratory
Wright-Patterson AFB, Ohio

12. REPORT DATE: August 1977

13. NUMBER OF PAGES: 85

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office):

15. SECURITY CLASS. (of this report): Unclassified

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE:

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Software
Cost Estimating Relationships
Cost Models
Life Cycle Costs
Operations and Support

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The cost analysts of today are not only faced with predicting or estimating the research, development, test, and evaluation (RDT&E), operation and/or support costs of weapon systems hardware, but are also expected to derive a similar cost estimate for software. This report reviews several software cost estimating "models" which exist today, and gives examples of their application to actual software programs. The models or cost estimating relationships (CERs) presented generally provide software cost estimates for the RDT&E

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. ABSTRACT (Cont'd)

phases of the life cycle, but have yet to be validated. The purpose of this report is to present some of the more popular software cost models and to encourage validation studies to determine which methods or models provide greatest promise for accuracy. It is hoped that this report will stimulate research and analysis into software cost estimating similar to the work being done in hardware cost prediction techniques.

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This report has been reviewed by the Information Office (OI) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

Thomas G. James Jr.

THOMAS G. JAMES, 1LT, USAF
Project Engineer

Richard L. Bush

RICHARD L. BUSH, Major, USAF
Avionics Synthesis & Analysis Branch
Systems Avionics Division

FOR THE COMMANDER

H. MARK GROVE, P.E.
Acting Chief
System Avionics Division

"If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify AFAL/AAA *,W-PAFB, OH 45433 to help us maintain a current mailing list".*

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

TABLE OF CONTENTS

TABLE OF CONTENTS (Cont)

LIST OF ILLUSTRATIONS

LIST OF TABLES

SECTION I

INTRODUCTION AND SUMMARY

In a recent USAF study of information processing requirements it was shown that in almost all applications, computer software was the major source of difficult problems, a major contributor to operational performance penalties, and potentially the largest source of life cycle cost. During the acquisition cycle of a weapon system, the military spends half of the total system acquisition cost on software (Reference 1). The Air Force spent between one and one and one-half billion dollars on software alone in 1972, which represented an expenditure of about four to five percent of the total Air Force budget (Reference 2). In comparison, the Air Force spends only $300 to $400 million per year on computer hardware. There have been large expenditures for software packages in the recent past, and yet it appears that software development costs are continually rising ($6 to $30 per line of code and upwards to $150 per line of code for very complex space systems) (Reference 2). In a July 1976 "Newsweek", it was stated that in the 1950's the rate of computer hardware costs to software costs was 4 to 1, compared to the present figure of 1 to 4, a complete turn-around in a little over 20 years (Reference 3). Decreasing hardware production costs and increasing personnel costs, are partly the explanation for this turn of events. Figure 1 is an estimated distribution of the total portion of the USAF budget spent on software. Figure 2 shows the relationship of hardware to software costs projected to the year 1985 (Reference 4).

In addition to the constantly rising cost for software development, software reliability, unresponsiveness, and indirect costs associated with slippages in software developments are of major concern to the USAF. A number of reports stress the fact that in software products acquired by the military, the quality or "relaibility" of the software produced is generally unacceptable (error rates of over 1 error per 100 lines of code (Reference 2). The CCIP-85 report (Reference 5) states that military software is extremely unreliable and unacceptable at the present time. A number of examples were given which indicate that software errors have

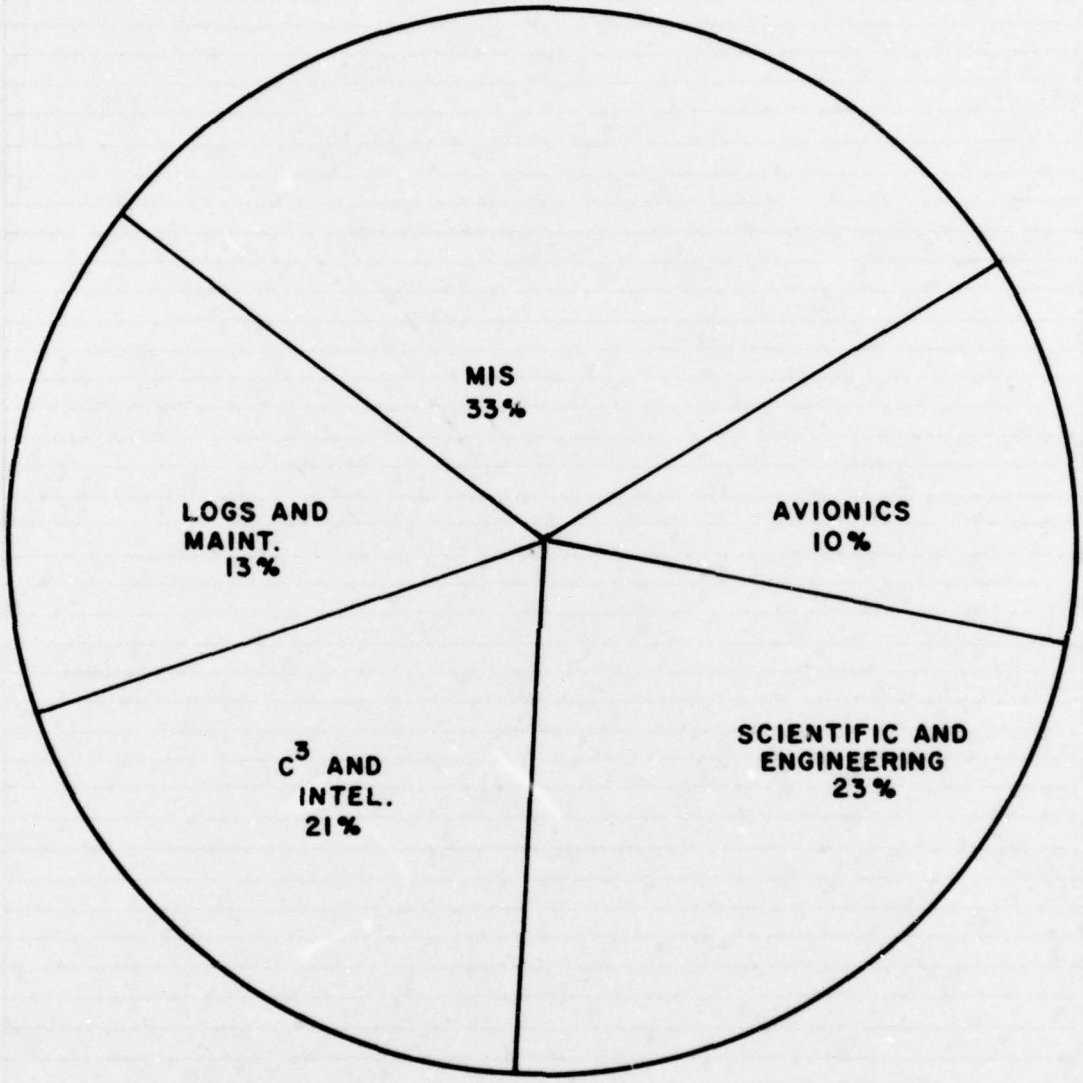


Figure 1. Estimated Distribution of USAF Software Costs (Reference 2)

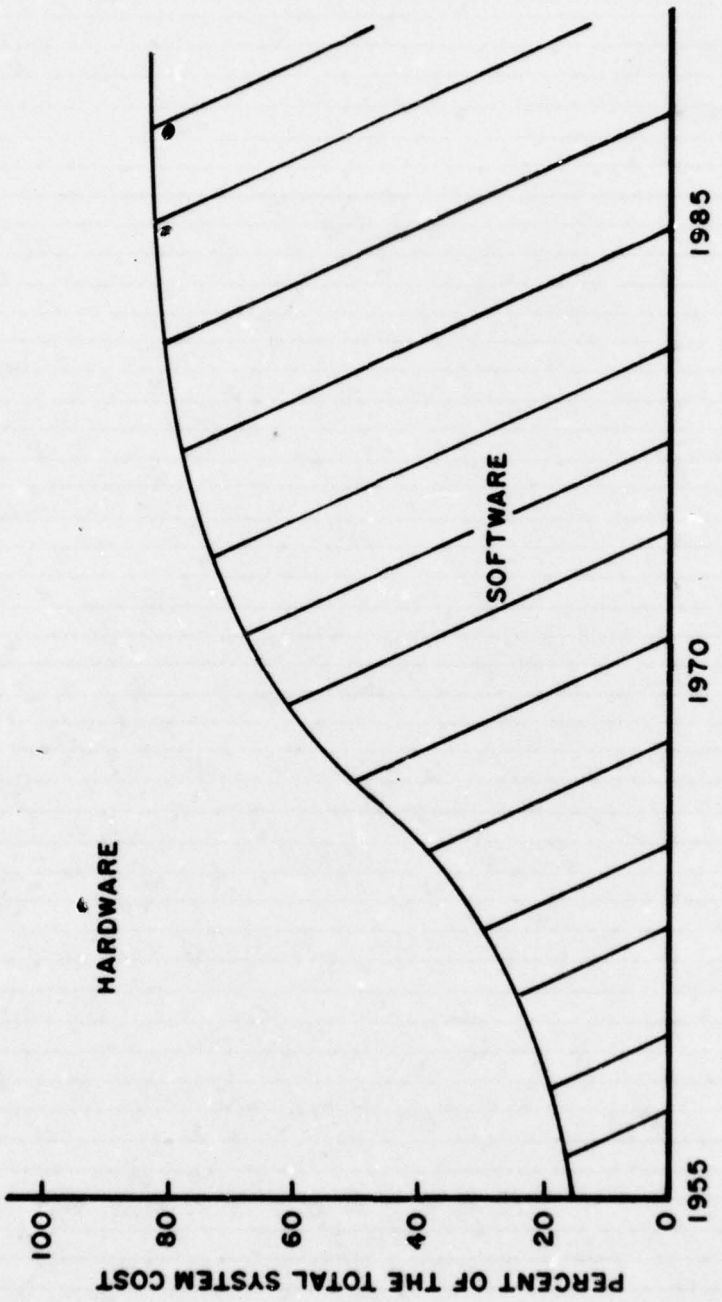


Figure 2. Hardware/Software Cost Trends (Reference 4)

caused considerable loss in terms of hardware equipment. Current Air Force software reliability problems indicate that software errors could cause the Air Force to lose critical command post or satellite capabilities in a strategic crisis situation.

Also, software is frequently unresponsive: 95 percent of the SAC Automated Command and Control System (465L) software package delivered to SAC had to be rewritten to meet SAC's operational needs and 67 percent of the Seek Data II software used during the Vietnam conflict had to be rewitten (Reference 6). In addition, it has been established that indirect costs of software slippages generally far exceed the direct costs (Reference 7).

The above examples emphasize the general criticality of military software, where many operations have to be performed in a few seconds or less. As a result, the military is taking a closer look at software development procedures and a greater portion of its R&D resources are being allocated to the software verification and validation (V&V) and development tools. However, in order to reduce the costs of software, the Air Force should have a defined method which will allow the analyst, software engineer, and/or project manager to estimate software development and support costs, given the basic requirements of the new system being developed. There exists today computerized models to predict the cost of hardware in terms of research and development, acquisition, and operation/support costs. These models, some of which have been validated, are widely used. (NOTE: There are those who will disagree that the military possesses the models to accurately predict hardware costs, but "life cycle cost" models for hardware do exist). Hardware has been in existence longer than software yet there is still no widely agreed-upon hardware life cycle cost methodology. Hence, the expectation that a validated/proven software life cycle cost model will come along in the next few years may not be realistic but it is certainly worth working toward.

The purpose of this report is to investigate the field of cost estimating as applied to computer software, primarily for the purpose of estimating the total life cycle cost of software for new avionics equipment under development. In this report past, present, and future efforts to derive a valid methodology to predict software life cycle costs will be discussed. Several methods or "models" which are usable today will be presented or referenced. The statistical confidence with which one may use the methods, however, is quite low. Therefore, they are presented not as tested, well-proven tools, but as guidelines to be used in conjunction with additional techniques, experience, and judgement. It is hoped that the reader will acquire some knowledge as to what is being done to predict software costs. The acquisition of software, the support of software, and the reliability of software are topics permitting unlimited discussions. This report will deal with each area in as much detail as time permits, primarily focusing on the acquisition costs of software. Due to limited time and space, the models are described in very concise terms in this report. Before actually applying a model, the reader is referred to the source document containing a more detailed discussion of a particular equation/model.

The following types of computer software programs were analyzed by various organizations in an attempt to derive a software cost estimating procedure: Management information systems (MIS); avionics; scientific and engineering; logistic and maintenance; command, control and communication, and intelligence. The question of whether avionics software is more costly than MIS software, etc., does not seem to be addressed to the point of defining a definite relationship, although it is thought that space and avionics software are more costly due to required testing and more detailed design effort.

SECTION II

DEFINITION OF TERMS

Most persons that will read this report are already familiar with the terms that have and will be used. Since, however, most readers have their own definition of terms such as software, reliability, operation and support, etc., these terms will be defined to insure a common foundation for the discussion to follow.

1. SOFTWARE

AFAL-TR-73-341 defines software as "the programs and routines used to extend the capability of automatic data processing equipment." To expand the definition of software as defined in AFAL-TR-73-341, this report also considers software as the programs and routines used to extend the capability of computers which are imbedded within weapon systems (not merely automatic data processing equipment). Software is further broken down into two types, basic software and application software. For purposes of this report, the term software will include all necessary documentation from functional specifications to flowcharts and users' manuals as well as the actual computer code.

a. Basic software comprises those routines and programs designed to extend or facilitate the use of particular automatic data processing equipment, the requirements for which take into account the design characteristics of such equipment. This software is usually provided by the original equipment manufacturers and is normally essential to and a part of the system configuration furnished by him. Examples of basic software are executive and operating programs, diagnostic programs, compilers, assemblers, utility routines, file management programs, and data management programs.

b. Application software consists of those routines and programs designed by or for automatic data processing equipment users, to accomplish specific mission-oriented tasks, jobs, or functions using the automatic data processing equipment and basic software available. Except for general purpose packages which are acquired directly from software vendors or from the original equipment manufacturers, this type of software is normally developed by the user in-house or through contract services.

2. REAL TIME SOFTWARE

A real time computer software system is defined as one which controls an environment by receiving data, processing them, and taking action or returning results sufficiently quickly, to effect the functioning of the environment at that time. "Sufficiently quickly" refers to the time which "allows users to interact with the computer on a time scale appropriate for human beings -- on the order of a few seconds between responses."

3. SOFTWARE ACQUISITION COST

The term acquisition cost of software is used in the same sense as acquisition cost of hardware. It includes the cost of analysis, design, programming, checkout, test, and documentation.

4. SOFTWARE RELIABILITY

Software reliability is the rate at which errors are detected in a program, i.e., number of errors per unit time of operations. A more formal definition of reliability states "Reliability - the characteristic of an item expressed by the probability that it will perform a required function under stated conditions for a stated period of time." (Reference 3). "Reliability is the measure of the frequency of failure of the computer software." (Reference 5).

5. MAINTAINABILITY

Maintainability is defined as a measure of the ease with which errors in a computer program can be corrected and system function and capability can be expanded or added. Unlike hardware, software maintainability entails some "redesign."

6. OPERATION AND SUPPORT

The operation/support (O&S) or maintenance costs of software include the costs associated with using or "running" the computer programs, modification or adaptation of an existing program to a computer system or to accommodate changes in system software, and the general day-to-day reprogramming that must be accomplished to keep the program operational. The operation/support costs are directly related to the reliability (number of errors) and the maintainability (cost to fix the errors).

7. LIFE CYCLE COST

The life cycle cost (LCC) of software is the total of the research and development, acquisition, and operation and maintenance costs.

SECTION III

REQUIREMENTS TO FORECAST SOFTWARE COSTS

The requirements to forecast software costs can be broken down into three areas of concern. The first and primary area is, what are the different methods available today for deriving a software cost estimate. Once the method has been determined, how are the historical data items collected to utilize the method. The method of deriving a software cost estimate plus the management responsibilities and proposed work breakdown structure for the collection of software data will be discussed in the following paragraphs.

1. METHODS FOR DERIVING SOFTWARE COSTS

In October 1974, a government/industry software workshop was held at Hanscom AFB, Massachusetts, sponsored by Electronic Systems Division (ESD) (AFSC). The purpose of the workshop was to "improve communications between industry and government in the problems of forecasting software development costs" (Reference 8). The objective of agencies dealing in software is to improve the accuracy/credibility of future software cost estimates for electronic defense systems. The workshop listed several methods for deriving software cost, the principal ones being factors, experts, ratio to previous experience, ratio to total system dollars, and probabilities.

The factors method involves identification of cost drivers and the formulation of an equation/series of equations relating these drivers to cost. These equations, cost estimating relationships, (CER's) are derived through the application of statistical methods to appropriate historical data. During the ESD Workshop the following dominant cost drivers were identified: (1) number of instructions in the program, (2) type of programming language (Higher Ordered Language (HOL), Machine Ordered Language (MOL)), (3) real time application, (4) type of program, (5) desired quality, (6) amount of documentation, (7) hardware constraints, (8) schedules, (9) size of data bases, (10) complexity, and (11) personnel

and management functions. Table 4 lists the driving factors the government and industry agreed upon and best guesses as to the effects these drivers have on software costs. NOTE: Items 1, 2, and 10 are common factors seen in the majority of CER's developed thus far. The factor method is good from the standpoint that it provides a quantitative relationship which is easy to apply. A major drawback of the factors method is that some inputs are subjective in nature, such as the complexity of the program, the skill level of programmer, etc. Development of reliable software cost estimating relationships (SCER's) using the factors method is currently limited by the quality/quantity of historical software cost data available. The data problem and what is being done to overcome it, will be discussed later.

The method of "experts" (delphi technique) has been used much in the past. This method, as the name implies, is dependent upon subjective opinions of a group of experts in the software field. Results are obviously only as good as the participants of the group. The method of experts, very similar to the corporate approach, is directed more along the lines of engineering estimates than statistically derived CER's. After estimating program size and complexity (usually by comparison with similar previous programs), historical corporate productivity data is applied to estimate direct labor hours (and thus direct costs) for coding and debugging. Costs for all other phases and factors in the development process are then estimated as (historical corporate) percentages of this direct labor cost.

The two methods of ratio to previous experience and ratio to total system dollars both have the same characteristics. Their good feature is that the ratio is developed on real experience and the drawback is the degree with which the results represent the actual cost incurred. In both ratio methods the data problem appears again; that is, there is the need for better data collection and analysis.

The difficulty of understanding the procedure used detracts from the probabilistic method. This method reflects reality by using past programs but it is harder to sell its use to someone with no understanding of probability theory. Again, to develop "good" best fit probability models, the need for data is ever present.

2. MANAGEMENT VISIBILITY

Management control, visibility of the software structure, and a standardized framework for collecting historical costs, sizing and requirement data, is greatly needed by the Air Force today. A point stressed during the ESD Workshop (Reference 8) is the need for good specifications early in the program. "A good specification is fundamental to building a realistic software cost estimate." Costing in the Air Force Avionics Laboratory and throughout the R&D community is done at various stages in the system life cycle, but the original most crucial costing is done in the conceptual phase. During this timeframe high level management often wants to be able to make a "go or no go" decision as to system development, usually based on a cost-benefit analysis (costs to develop, deploy, and maintain the system versus the benefits to be gained by acquisition of the system). In these early stages the costing study is extremely difficult because system documentations are usually incomplete or lacking in detail and may be inconsistent or ambiguous. The appropriate time at which an initial software cost estimate should be attempted is at the earliest possible point after functional design specifications are completed (Part I Specifications). Once the functions and/or modes the software program will deal with are known, the "size of your program" and a "rough" estimate of the software cost can be developed from analysis of the functions of the program. The cost of deriving a good software cost estimate is high since much preliminary design work is required. However, the software workshop agreed that in order to accurately predict software costs, a considerable amount of design work and project planning must have been accomplished.

According to industry comments, the majority of software cost estimates are obtained from elementary sizing parameters of the "estimated number of instructions" usually derived from historical experience and/or engineering judgement. Once the size of the program is determined, then manpower requirements are estimated, leading to the cost of the software package development. Both rule of the thumb and mathematical methods do exist but none are very reliable or validated.

3. WORK BREAKDOWN STRUCTURE

A major problem in the process of comparing or tracking actual software development cost is that there is a lack of a common language, methodology, and work breakdown structure (WBS) which would provide a basis for developing and comparing cost estimates. In a report prepared for NAVSEA, "Interim Guidance for Preparation of Cost Estimates for Tactical Software Programs," Oct 74, an interim work breakdown structure was provided as the frame-of-reference for all NAVSEA tactical software program cost estimation (Reference 9). The report provided a brief description of a typical software development process and how the various activities relate to the work breakdown structure. Worksheets or summary report formats were presented to cover each of the following topics:

a. Cost Estimate for XXX Tactical Software.
b. Cost Incurred Schedule.
c. Tactical Software Program Summary
d. Milestone/Resource Allocation

The problem and the inability to apply WBS methods to software development can best be summed up in a quote from the NAVSEA Report: "In the past, industry and NAVSEA project managers/engineers have not been able to describe or define the software programs for a tactical system at levels parallel to that which have been developed for technical management of hardware. This inability to develop realistic work packages and milestones for management of software programs has resulted in ineffective monitoring and cost forecating. In addition, the lack of a suitable common structure of WBS language has limited the development of norms and valid data banks for transferring experience and interfacing with industry." (Reference 9)

It is quite possible that the approach of the NAVSEA report has been adopted in MIL-STD-881, but it appears that the WBS has not yet been fully utilized by managers of software packages, which leaves the cost estimators in a state of flux.

3. DISTRIBUTION OF SOFTWARE DEVELOPMENT EFFORT

A majority of the models presented in Section V of this report arrive at a total cost for software development. A distribution of software development effort or allocation of effort during this phase is sometimes desired. The development process can be broken down into three major phases: Analysis and Design, Coding and Debugging, and Integration and Test. Documentation costs will be included in Integration and Test. Table 1 represents some findings into how the three phases are distributed as a percent of the development effort. A general consensus of a 40, 20, 40 percent distribution can be drawn (i.e., 40% for Analysis & Design, 20% for Coding and Debugging, and 40% for Integration and Test). If only airborne and space programs are considered, it can be seen that more emphasis is being given to Analysis and Design and a great deal to Integration and Test. In airborne programs, coding and debugging are a smaller part or percent of the effort.

TABLE 1

DISTRIBUTION OF SOFTWARE DEVELOPMENT EFFORT

SOURCE	PROGRAM/COMPANY	ANALYSIS AND DESIGN	PERCENTAGE AND DEBUGGING	INTEGRATION AND TEST
Ref 2	SAGE/NTDS	35 %	17 %	48 %
Ref 2	TRW(COMMAND/CONTROL)	46	20	34
Ref 2	GEMINI/SATURN	34	20	46
Ref 2	OS/360	33	17	50
Ref 2	TRW (SCIENTIFIC)	44	26	30
Ref 10	RAYTHEON (BUSINESS)	44	28	28
Ref 10	INFORMATIES CORP	46	16	48
Ref 11	TITAN III	33	28	39
Ref 11	X-15	36	17	47
Ref 11	APOLLO	31	36	33
Ref 11	GEMINI	36	17	47
Ref 11	SATURN V	32	24	44
Ref 11	AIRBORNE DAIS (EST.)	38	15	47
Ref 11	GRC EXPERIENCE	30	20	50
Ref 11	SKYLAB	38	17	45
Ref 11	TRW	40	20	40
Ref 11	SETS/BL	42	18	40
Ref 12	IBM	30	40	30
Ref 12	AEGIS	38	26	36
Ref 12	AN/BQQ-5	31	43	26
Ref 13	COST-BY-FUNCTION MODEL	34.5	18.0	47.5
	AFAL (AAA-3)	38.7	21.7	39.6
AVERAGE		36.8%	22.9%	40.7%

SECTION IV

PAST/PRESENT AND FUTURE STUDIES (STATE OF THE ART)

The alarming increase in software development costs and the decreasing dollar resources available for software development have forced the government and especially the military to find new ways of producing quality software within very limiting constraints. These costs, coupled with the fact that the quality of software produced is generally unacceptable (error rates of over 1 error/100 lines of code), has spurred several Air Force organizations into conducting various studies on software cost estimating procedures and development methods.

1. ROME AIR DEVELOPMENT CENTER (RADC) EFFORTS

The Rome Air Development Center has begun an extensive study of the software development field. The goal of the RADC program is to achieve higher quality, lower priced software through the reduction of intrinsic error rates; the improvement of programmer productivity (by using better programming languages, better design, coding and testing techniques, and better management control) and improvement of the readability, portability, documentation, and maintainability of software code.

To achieve the Air Force objectives in the software development area, the RADC approach is to develop a system of software facilities, all linked to a centralized software data base. Three facility development efforts are currently under negotiation at RADC for facilitating development of the automated analysis system. The first is a multiregression facility for statistically correlating various data with respect to reliability, cost, and productivity. The second is a facility to use RADC's On-line Pattern Analysis and Recognition System (OLPARS) to use the pattern recognition for determining software reliability. The third is a language control facility for collecting various information on language usage and relating errors to specific language constraints.

Principal functions of the centralized software data base will be to: (1) collect software production data, (2) provide a computerized data base of raw data for reliability, cost and productivity analysis, (3) provide analytical tools for data analysis and modeling, and (4) generate standard reports on information contained in the repository. The necessity for collecting software data is based on the fact that no clear conclusions or predictions can be made about quantities of interest, such as the reliability of software or an accurate estimation of software production costs, without the historical data base. The repository data related to the design, coding, testing, and maintenance of software will include software error data, software cost, complexity and productivity data, and computer language. Once centralized software data bases have been established, a number of efforts can proceed. For example, information on the cost of software developments as related to the techniques used to develop the software and on error rates as a function of a particular language or a language feature are important areas of investigation that will be aided significantly by the repository.

A major requirement for effective use of the software data base is a software modeling facility. This type of facility would enable researchers to model various aspects of the software development process. For example, cost estimating models based on factors like functional requirements, problem complexity, and programming experience, and sizing models for determining necessary computer resources, are critical for accurate software cost estimates. Reliability models for predicting the occurrence or number of software errors in operational software are important in answering questions related to the release of software for operational use and the progress of software testing. These models, plus software complexity and production models, can lead to a true appreciation of the significant factors underlying software development, thus leading to increased control over the software process. Contractors involved in RADC software programs include System Development Corporation (SDC), Illinois Institute of Technology Research Institute (IITRI), Polytechnic Institute of New York (PINY), and the MITRE Corporation.

The System Development Corporation (SDC) is conducting a data collection study for RADC. SDC is studying the general area of data collection, storage, and retrieval. Major emphasis has been placed on the problems related to the collection of software production data. They also analyzed previous efforts by IBM, TRW, and MITRE (Reference 1) in the software data collection area.

The Illinois Institute of Technology Research Institute (IITRI) is under contract with RADC to investigate, among other things, the specifications for a pilot repository facility at RADC. Thus, SDC and IITRI are working closely to develop specifications for a Software Data Collection & Repository System at RADC.

The MITRE Corporation has developed models for measuring structural complexity of software, and is currently developing a Software Implementation Monitor (SIMON) for RADC to automatically gather and analyze data during software development.

Polytechnic Institute of New York (PINY) is currently investigating modeling techniques, similar to those used for hardware reliability, for use in the software area. For example, PINY has developed a model for predicting the reliability of software based on the use of Markovian processes to determine the probability that a given software system is in either an "up" state (no errors present in the system) or a "down" state (an error has occurred and is being corrected). Other areas which PINY is investigating at this time include:

a. Models to measure software complexity and develop relationships among error content, debugging effort, program size and run time.

b. A study of the effects of modular and structured programming on program errors.

c. Models to test effectiveness in removing software errors.

d. Development of models for comparison of different programming languages with respect to such features as core size, run time, development test, and debugging costs.

With respect to software reliability/maintainability, RADC is currently planning an effort to develop software reliability models based on the Bayesian statistical theory. Along with this, methods will be developed for making acceptance or rejection decisions about a software package during software testing using these Bayesian models and Bayesian techniques. Also planned by RADC is the evaluation of computer programs and designs in terms of a quantitative measure of maintainability, and the restructuring of computer programs for reliability and maintainability improvement. As part of this effort, a maintainability model will be developed which tracks and measures the propagation of modifications and/or errors through a system of software modules, thus leading to a measurement of maintainability.

2. ELECTRONIC SYSTEMS DIVISION (ESD) EFFORTS

Two Electronic System Division (ESD) organizations, ESD/MCIO and ESD/ACCI, are conducting studies analyzing the software development cycle and predicting software costs.

On 1-2 October 1974 an ESD workshop entitled, Government/Industry Software Seizing and Costing was held at Hanscom Air Force Base, Bedford, Massachusetts. The primary output of the workshop was a list of factors and details on how these factors affect the cost of software development. This list is what is referred to in this report as the "ESD Model", and is described in detail in Section V-3. The dominant factors affecting software costs were identified as the number of instruction, programming language, real time application, type of program, desired quality, amount of documentation, hardware constraints, schedules, size of data base, complexity, stability of requirements, and personnel and management required. Certain procurement procedures such as subordination of software design goals to hardware design goals also were identified as having a decided effect on software costs. The workshop agreed upon one general point, i.e., that deriving a good software cost estimate is very expensive.

Another point of interest was the need for an improved work breakdown structure (WBS) and the fact that MIl-STD-881 has not been fully utilized to decompose complex software projects into manageable work packages.

To further the software cost estimating techniques that exist today, ESD/MCI has two major efforts at the present time (Reference 6). The first is a study entitled "Life Cycle Costing of System Software/Computer Resources," being conducted by General Research Corporation (GRC). The objective of the study is to develop WBS down to a level sufficient to identify software cost elements and functional requirements. The second step of this effort will be to collect data against this WBS. After these two steps have been taken, GRC will employ statistical methods to develop CERs relating previously identified cost elements to resource expenditure for each phase of the software life cycle. A follow on effort entitled "Software Cost Prediction Aids" will render these CERs compatible with a generalized life cycle cost model like the MITRE Electronic Systems Cost Model, or a functional description tool like the Computer Aided Requirements Analyses (Reference 14). Development of a total life cycle cost model for software is expected to be completed in October 1976. The results of the GRC effort under contract F19628-76-C-0180 are documented in a preliminary draft report entitled "Cost Reporting Elements and Activity Cost Tradeoffs for Defense System Software." The six month study investigated the problems of software cost estimation, hypothesizing relationships, gathering and analyzing data, and examining reporting systems. There exists equations relating cost (in terms of estimating the man-months) to the different phases of the software life cycle. The equations are not presented here since it would be worthwhile to wait for the final report to be released. The following quote exhibits the confidence of the derived relationships: "Our second major objective was to develop improved software cost estimating relationships. A significant amount of work had been previously devoted to this task. The work was performed by competent groups and focused on estimating total man-hours or costs. Results have been disappointing, with derived relationships exhibiting large variance." Some of the major findings by GRC include the following.

a. Accurate estimating relationships for each life cycle phase cannot be developed independent of the other phases

b. Estimating the tradeoffs between the life cycle phases is of prime importance.

c. Estimating the tradeoffs can also lead to the development of rules for optimal allocation among life cycle phases.

The GRC study to date is the most current and complete effort accomplished to determine operation and support cost of software.

The second major effort is the Air Force Software Library. ESD/MCI has developed a software library which is presently on-line in prototype form at ASD? The library is designed to collect technical data on existing software packages. The library at the present time contains a description of the software program and the person or organization to contact for additional information. An effort is now underway to collect and store data, where available, on resources expended in developing and maintaining these programs.

In a draft of a proposed in-house research program for improving software cost estimating, ESD/ACCI proposes to develop two models, one a "robust regression model", the second a "software development simulator" (Reference 14).

The robust regression model will be developed to handle small sample sizes. AFSCM 173-1 describes the ground rules and methods of utilizing the linear statistical model as a standard technique for developing estimating relationships. AFSCN 173-1 adopts the principle of least squares which is based on possessing a normal or "bell-shaped" statistical distribution. When the variations are not distributed normally, these properties cannot be proven to be true. Furthermore, AFSCM 173-1 states that "when sufficient data points are available, the distribution of sample means will remain normal to a satisfactory degree of approximation." According to the ESD/ACCI draft report, 30 data points are generally a large enough sample for the normality assumption to be sufficiently approximated. In the area of software cost estimating, data definition and collection problems do not generally give us 30 data points, (homogenous

sample sizes). "Thus, there is no reason to believe, a priori, that standard statistical techniques will produce accurate CER's, and the failures of SDC, GRC, and Tecolati are not all that surprising." (Reference 14) The "robust regression technique" suggested by Capt Bourdon (Reference 14) to handle this data problem is to employ the Kurtosis (fourth standardized moment) of the least squares residuals. This method has been demonstrated for the simple, two-variable model and quite possibly can be applied to a multivariate model. The Kurtosis then can be used when random variables are not known to be distributed normally. According to Capt. Bourdon, this method is as good or better than least squares regression down to sample size of four.

The second model proposed is the software development simulator which would employ a Monte Carlo sampling technique. The simulator envisioned is predicated on the hypothesis that cost is an explicit function of the time usage of direct labor and computer hours. An estimation of the labor and computer hours consumed as a function of time can be made and in turn the cost arrived at by applying standard factors for engineering management, overhead, etc. "Thus the simulator serves to generate and tabulate statistics on the consumption of labor and computer resources by simulating the software development process." (Reference 14)

ESD/ACC pointed out in Reference 14 that there is a definite need for an ad hoc planning group to guide future research in the software cost estimating area, to eliminate duplication of research dollars spent and time and effort devoted to similar tasks. ESD/ACC recommends that they be designated the software cost estimating research focal point for all activities in the area conducted by agencies under the operational control of the ESD Commander. The proposed ad hoc planning group would be responsible for:

a. Defining, monitoring, and reporting on the progress of all in-house and contracted efforts aimed at bettering the ability to predict the cost of software.

b. Coordinating on all Statements of Work and serving on all source selection boards for procurements associated with software cost estimating methodologies.

c. Organizing and coordinating periodic symposia and seminars for the exchange of data, ideas, and findings between government, industrial, and academic institutions active in software development and cost estimating.

3. TRW EFFORTS

TRW is currently investigating the types of errors which are made most frequently and an error classification scheme. TRW is also analyzing how personnel, hardware problems, hardware interfaces, operational timing, and input requirements contributed to errors which occurred. TRW has developed a Mathematical Theory of Software Reliability (MTSR) for predicting the reliability of software systems based on the complete set of possible data input, values, and the logical structure of the component modules. They are currently applying this theory to an actual software development project and are further analyzing the theory to determine the effects of errors removal on reliability and the variance in sampling techniques for measuring the reliability.

4. RCA "PRICE" SOFTWARE ACQUISITION MODELS

RCA is currently developing a software acquisition cost model analogous to the proprietary hardware acquisition model. At the present time, numerous DoD and contractor organizations utilize the PRICE model to develop cost-estimates for new hardware equipment/systems, in terms of development and acquisition costs. Business Week, 7 June 1976, "RCA's Uncanny System for Estimating Costs" states that "Altogether, Systems Command will spend $200,000 on PRICE this year. Some Air Force procurement agencies even require that bidders' proposals include data in the specific form need for PRICE input." Over the past two or three years, RCA has been working to develop a software cost estimating technique that is similar to the RCA PRICE model. It would be unfair to the reader to try to predict the format, or the accuracy of the model at this time, but based on the

hardware model, PRICE, it is safe to state that this model looks promising with respect to prediction of software development costs. The extent to which the new software cost estimating model will be used depends upon the cost charged the user by RCA and validation results. The model should appear on the market for use sometime during the summer of 1977.

SECTION V

SOME SOFTWARE DEVELOPMENT COST ESTIMATING MODELS

In this section, several software cost estimating models are presented. It is generally agreed that no software cost estimating relationship (SCER) or model has been adequately validated. Hence, the use of these models must be viewed in this light. Section VI includes demonstrations of the use of each of these models on actual software development programs but it should not be viewed as a validation or even an evaluation study. Actual costs of software development programs required for such an evaluation were not available. (Only two computer programs analyzed have actual costs associated with them). The majority of models considered are based upon an initial estimate of the number of instructions to be written (sometimes arrived at by estimation of the number of functions of a software program and general information as to number of instructions per function). This implies that even if the SCER were to have say an $r^2 = 0.95$, the number of instructions estimated or "guessed" drives the SCER output total cost. (NOTE: r^2 refers to the coefficient of determination which is a measure of dispersion showing the proportion of total variance accounted for by the estimating relationship). Another general observation about the models is the fact that most were developed on relatively small data bases (as small as two programs and as large as 169 programs).

1. THE WOLVERTON MODEL

In November 1973, Ray W. Wolverton of TRW Systems Group presented a paper entitled "The Cost of Developing Large Scale Software" (Reference 10). This, most probably, was not the first attempt at a software cost estimating model, but has become the most widely referenced work on software cost estimation. The Wolverton model is the most widely used and accepted software cost estimating technique developed thus far. This methodology is applicable to large scale software development programs which utilize a "structured programming" design approach. Structured programming implies modular form.

The basis for the model is a TRW proprietary data base containing historical information in the form of cost per instruction. Wolverton assumes that the development cost varies proportionately with the number of instructions. For each identified routine, the procedure combines a user supplied estimate of the number of object instructions, category, and relative degree of difficulty with relationships based on the historical data base to determine a trial estimate of the total software development cost.

The first step in the procedure is to estimate the number of instructions in each category. The categories which Wolverton defines are as follows:

a. (C) Control routine, which controls execution flow and is non-time critical.

b. (I) Input/output routine, which transfers data into or out of the computer.

c. (P) Pre- or post-algorithm processor, which manipulates data for subsequent processing or output.

d. (A) Algorithm, which performs logical or mathematical operations.

e. (D) Data management routine, which manages data transfer within the computer.

f. (T) Time critical processor, which is a highly optimized machine dependent code.

To obtain a relative degree of difficulty there are basically two substeps involved. First, determination of whether or not the routine is an "old" or a "new" program. Once that determination has been made (old or new), then the program must be classified as to whether it is an

easy, medium, or a hard program to code/design. Therefore, the possible degrees of difficulty are:

Program	Easy	Medium	Hard
Old	OE	OM	OH
New	NE	NM	NH

where O = old, N = new, E = easy, M = medium, and H = Hard. The results of step one are then multiplied by the significant cost per instruction (CPI) expected for the type and difficulty categories. The total expected cost of the program is the sum of the above calculations. Table 2 is a breakdown of the Category/Type and the cost per word expected. As Capt. Gaumer pointed out in his thesis, the costs associated with Wolverton's categories were extracted from actual historical costs incurred by TRW, Inc. based on 1972 dollars (Reference 15). Using Capt. Gaumer's method and the Implicit Price Deflators index listed in the "Survey of Current Business", Wolverton's 1972 figures are multiplied by an inflation factor of 1.27. Hence, the figures in Table 2 are in 1976 dollars.

TABLE 2

CATEGORY/TYPE VS TYPE PER WORD

OLD	C	I	P	A	D	T
E	$27	$23	$22	$19	$30	$95
M	34	30	29	25	39	95
H	38	34	33	28	44	95
NEW	**C**	**I**	**P**	**A**	**D**	**T**
E	$42	$36	$36	$30	$47	$95
M	51	44	43	38	58	95
H	62	55	53	44	71	95

The major pitfall with the Wolverton model lies in the initial estimation of the numbers of instructions by degree of difficulty and category. Once these estimates are obtained the model is easily applied. Results naturally depend on the accuracy of the initial estimates.

2. MODIFIED WOLVERTON MODEL

The System Evaluation Group, of the Air Force Avionics Laboratory developed a computerized version of the Wolverton Model for rapid analysis of software development costs. As the title suggests, this model is based on the TRW work conducted by Ray Wolverton.

The only required input to the computer program is the number of instructions by type (i.e., number of C,I,P,A,D, and T instructions) as defined in Section V-1. The program utilizes ten equations to obtain the cost per instruction for each type. These equations were obtained through regression analysis using the data displayed in Figure 12 of Reference 10. The cost for time critical processor type (T) instructions is assumed constant as in the Wolverton Model. Costs associated with the level of effort are computed as follows: (1) total cost of the program is calculated from the number of instructions and cost per instruction by type; (2) analysis is 20 percent of total cost, design is 18.7 percent, coding is 21.7 percent, testing is 28.3 percent and documentation is 11.3 percent.

The Modified Wolverton Computer Program generates program development costs for "new" and "old" code, for programs ranging in "percent difficulty" from 10-90 percent. The user must, based on subjective decisions relative to these characterizations, select the appropriate cost figure from the spectrum of data generated. For the Modified Wolverton Model, Wolverton's categories of easy, medium, and hard are redefined as "percent difficulty" on a scale of 10 to 90 percent. The relationship between these categorizations is presented in Table 3 below.

TABLE 3. PERCENT DIFFICULTY

Easy	Medium	Hard
10-20-30%	40-50-60%	70-80-90%

The following characterizations of easy, medium, and hard programming tasks developed by IBM may assist the user in assigning "percent difficulty" figures when utilizing the Wolverton or modified Wolverton models.

a. Easy: Very few interactions with other system elements. The class includes most problem programs or "application" programs. Any program whose main function is to solve mathematical or logical problems is probably in this class. Easy programs generally interact only with input/output programs, data management programs, and monitor programs.

b. Medium: Some interactions with other elements. In this category are most utilities, language compilers, schedules, input/output packages, and data management packages. These programs interact with hardware functions, with problem programs, with monitor, and with others in this class. Complicated by being generalized enough to handle multiple situations: I/O from many different I/O devices or management of class files with variable number of indices.

c. Hard: Many interactions with other system elements. All monitors and operating systems fall into this class because they interact with everything. Special purpose programs, such as a conversational message processor, may be in this class if they modify the master operative system.

The Modified Wolverton computer program listing and sample inputs and outputs are contained in Appendix A.

3. ESD MODEL

The summary notes of the October 1974 Electronic Systems Division sponsored software workshop (Reference 8) form the basis for what is referred to herein as the "ESD Model". Factors identified as impacting software costs are provided in Table 4.

TABLE 4

FACTORS INFLUENCING SOFTWARE COST

FACTOR	RELATION TO COST
Number of delivered source instructions	Linear, modified by other factors
Language	HOL: $6-12/source instruction
	MOL: $12-24/source instruction
Real-time application	RT: $30-60/source instruction
Type (OS, application, utility)	If OS, multiply by 2.5
Point on learning curve	If unfamiliar, multiply by 1.5 - 2.0
Application area (MIS, avionics,...)	Sometimes, as percentage of total system cost
	"Man-rated": test cost ∿ 40% of total
	Non-"man-rated": test cost ∿ 15% of total
Turnaround time	Approximately linear relation to testing cost
Amount of documentation	Approximately 10% of total; $35 - 150/ non-automated page
Hardware constraints	Asmptotic
Schedule realism	Percent added cost = percent of schedule acceleration
Amount of previous software used	Breakout and subjective
Size, structure of data base	Subjective
Complexity	Subjective
Stability of requirements	Subjective
Stability of development environment	Subjective
Representativeness of development environment	Subjective
Personnel	Subjective; approximately 5:1 variability
Development methods (e.g., structured programming)	Subjective; systematic approaches cheaper
Management	Subjective: high variability

The primary step in using this model is the determination of the number of delivered executable source instructions where delivered implies designed, integrated, tested, and document (Reference 8). Source instructions which for this discussion exclude comment cards, is considered a better estimation factor than the number of object instructions which is then used in the Wolverton and Modified Wolverton models.

Once the number of instructions and the language are known, cost factors presented in Table 4 are used to arrive at the basic cost figure. As can be seen from the table, many factors affect the cost estimate, such as whether it is a real-time application program, familiar, or unfamiliar program, etc.

The "relation to cost" for several of the factors identified as invluencing software cost are listed as "subjective". The size and structure of the data base is an extremely important parameter. Quite naturally, the effect on cost is more for large data file oriented projects but as of yet, no quantitative relationship similar to those developed for cost-per-instruction has been established. The complexity factor as of yet has not been defined in a way so as to be used reliably in a cost formula. Attempts have been made to correlate costs with such factors as number of interfaces, percentage of branch statements, and number of paths through a program, but without any highly reliable correlations. The effect on cost that the development environment has is merely the added cost required to adapt software to actual operational conditions such as different computer configuration and operating procedures; can be quite significant, upwards to 95 percent in some instances, but can only be estimated subjectively.

Quality of personnel is considered by many experienced estimators to be the most important factor affecting software development costs. Productivity variations of 5:1 between individuals are common. Yet to be developed is the quantitative effects on cost of using development techniques such as structured programming, top-down development, chief programmer teams, and automated aids. It is agreed that systematic

approaches to software development are better than disorganized ones. Possible payoffs for the use of systematic software development techniques are in operation and maintenance costs because of ease of debugging and rebuilding the program.

To sum up the ESD approach, the basic cost is arrived at by utilizing the number of instructions times the cost per instruction and adding cost for type of program, unfamiliar, real-time, etc. Subjective factors are then applied to adjust cost to reflect the development technique, personel, etc.

4. THE TECOLOTE MODEL

In this report, the Tecolote Model refers to the basic equations extracted from a report entitled, "A Provisional Model for Estimating Computer Program Development Costs," Dec 1974, (Reference 16) prepared by Brad C. Frederic of Tecolote Research, Inc., Santa Barbara, California, for the Resource Analysis Branch, Office of the Chief of Naval Operations, Department of the Navy specifically for estimating development cost for tactical software. Tactical software is defined by Frederic as any complete set of computer programs that resides in and drives a computer system within a fire control system. Mr. Frederic stressed the point that the model was a "provisional model," that is, serving only for the time being.

The report emphasized the problem of obtaining data to perform statistical analysis and noted that three large software cost data bases had been already compiled at System Development Corporation (SDC), TRW, and North American Autonetics (NAA). There were problems in the data collected by Tecolote (387 separate points from 15 source references) that proved insurmountable. Since the data had to be collected from rather outdated published sources, locating spokesmen familiar with the program to interpret the data was impossible. Therefore, the data base could not be treated or rationalized into a homogeneous base. Hence, Tecolote elected to undertake a small sample approach (5 data points) utilizing only data which they thoroughly understood, and where "the

estimating relationships developed would be more in the nature of engineering scaling laws than strictly derived statistical equations." (Reference 16)

The Tecolote analysis of software development included the following activities, as given by Wolverton (Section V-1):

a. Software requirements generation.

b. Preliminary software design (and release).

c. Detailed software design (and release).

d. Code and debug.

e. Development testing.

f. Validation testing.

g. Operation demonstration (and handover).

The types of computer architectures which this study included were single Central Processor Units (CPU), democratic, and autocratic. Single CPU involves a single central processor with storage and peripherals. The democratic architectures consider multiple CPU's operating in parallel with pairwise communication, common storage, and peripherals. Autocratic is a combination of a single CPU's and democratic subsystems acting in parallel, under the control of a separate single CPU executive.

Mr. Frederic noted that computer system speed and fast storage capacity are the major drivers of software requirements. The size of the program in this model is the number of machine language instructions. The size can be input as either the number of operational instructions or the number of delivered instructions. In general, the number of delivered is greater than the number of operational instructions. Operational instructions are those produced during development that are eventually installed in the tactical hardware; delivered instructions are all those instructions produced during development. The instructions contained in a development "test bed" which simulates hardware interfaces are an example of delivered instructions which never become operational. According to Frederic, the

number of operational instructions increases for tactical software directly as either the number of targets terminal-tracked increases or as the target approach speed increases.

There are five basic cost estimating equations derived by Tecolote given in Table 5. Each equation requires the input of one of five self explanatory variables. The equation which the user utilizes depends on the input variable which he is more confident about. For example, if the user knows the number of delivered instructions (D) then the equation $0.01(D)^{1.18}$, D in thousands, results in total development cost. Likewise, if the user knows number of operating instructions (O) the equation $0.01(O)^{1.24}$ gives you total development costs. Notes A, B, and C are helpful in terms of understanding the basic assumptions of the CER. The output is the total development cost in FY73 millions of dollars.

5. THE IBM MODEL

The IBM Model is documented in the IBM proprietary report "Estimating Software Life Cycle Costs: by John C. Malone, April 1975 (Reference 17). The report utilized software cost data which was derived from software projects performed by IBM, which, (1) employed top-down structured programming techniques and (2) utilized the Chief Programmer Teams Operation Concept. Structured programming techniques feature a simple flow of logic such that the program can be easily read and understood. Structured programming tends to improve both software reliability and maintainability but may not be efficient in terms of computer resource usage. Structured programming constrains the implementer to three basic constructs, "the straight line," "if then else," "do while (loop)." Top-down programming is starting development with the top module such that the real driver is used to test all submodules estimating interface problems.

TABLE 5

SUMMARY OF PROVISIONAL SOFTWARE ESTIMATING RELATIONSHIPS (SEE NOTE A)

INPUTS / OUTPUTS	M, TOTAL MAN-MONTHS LABOR	D, TOTAL DELIVERED INSTRUCTIONS (THOUSANDS)	O, TOTAL OPERATING INSTRUCTIONS (THOUSANDS)	AIR THREATS (B)		SEA THREATS (C)	
				S, TOTAL WORDS FAST STORAGE (THOUSANDS)	T, TARGETS TERMINAL-TRACKED	S, TOTAL WORDS FAST STORAGE (THOUSANDS)	T, TARGETS TERMINAL TRACKED
TOTAL DEVELOPMENT COST FY 73 $M	$0.0043(M)$	$0.01(D)^{1.18}$	$0.01(O)^{1.24}$	$0.0026(S)^{1.79}$	$0.30(T)^{1.88}$	$0.0043(S)^{1.79}$	$0.19(T)^{1.88}$
TOTAL MAN-MONTHS LABOR		$2.43(D)^{1.18}$	$2.52(O)^{1.24}$	$0.59(S)^{1.79}$	$69(T)^{1.88}$	$1.01(S)^{1.79}$	$45(T)^{1.88}$
TOTAL DELIVERED INSTRUCTIONS (THOUSANDS)			$1.03(O)^{1.05}$	$0.30(S)^{1.51}$	$17(T)^{1.59}$	$0.48(S)^{1.51}$	$.12(T)^{1.59}$
TOTAL OPERATING INSTRUCTIONS (THOUSANDS)				$0.31(S)^{1.44}$	$14(T)^{1.51}$	$0.48(S)^{1.44}$	$10(T)^{1.51}$
TOTAL WORDS FAST STORAGE (THOUSANDS)					$14.30(T)^{1.05}$		$8.30(T)^{1.05}$

NOTES: (A) COST ASSUME $3,930 FOR LABOR, $77/COMPUTER HOUR, AND 4.23 COMPUTER HOURS/MM.

(B) USE AIR THREAT COLUMN IF MAXIMUM THREAT APPROACH SPEED IS IN THE 250-700 M/SEC RANGE.

(C) USE SEA THREAT COLUMN IF MAXIMUM THREAT APPROACH SPEED IS 50 M/SEC OR LESS.

The chief programmer approach depends on top-down implementation, and matches personnel capability with the complexity of the modules they are to develop, i.e., the top-most complex modules are produced by a highly qualified software system specialist, referred to as the chief programmer. Less qualified personnel implement the lower lever modules under the control and guidance of the chief programmer. The chief programmer approach to software implementation is a good concept, but the staffing profile can make it difficult to employ. This model addresses only the software development phase. The data included costs for the development phase of both real-time and support software. The equations being of a proprietary nature could not be presented, however, the results of applying the model are presented in Section VI of this report.

6. NAVAL AIR DEVELOPMENT CENTER MODEL

The cost relationship (CER) discussed in this section was taken from a study done by Naval Air Development Center (NAVAIRDEVCEN or NADC) entitled, "A Cost By Function Model for Avionic Computer Systems", March 1971 (Reference 13). The NAVAIRDEVCEN developed an overall CER, comprised of several equations, which could be used for predicting total acquisition costs for research, development, test and evaluation, and production of future avionic computer systems. Reference 13 gives a complete computer listing of the "Cost-by-Function" model with its 10 basic modules. These 10 basic modules are as follows:

(1) Raw Technical System Requirements: Functional requirements of the system are translated by a function/structure requirements matrix to six variables denoting the raw technical requirements of the system.

(2) Total Technical System Requirements: The raw technical requirements are modified using system architecture factors to reflect performance needed.

(3) Modularized Technical System Requirements: Converts from total technical system acquirements to integral units of the selected hardware modules.

(4) Cost Trends Near Baseline: Cost trends with technical requirements are determined in the vicinity of each baseline. This module automatically recalibrates the model when new data becomes available.

(5) Programming Costs (RDT&E): The software requirements implied by module two are converted to RDT&E programming costs.

(6) Estimated Hardware Costs: RDT&E and First Unit Production: By utilizing the system performance characteristics, baseline characteristics and cost trends, the hardware costs are estimated. The model selects a baseline approximating the desired system.

(7) Production Cost Breakout by Year: A learning curve and quantity discount are employed and aggregated on a yearly basis via an input production schedule.

(8) Breakout of RDT&E Hardware Costs by Line Item: The results of module six are broken down by major line item.

(9) Breakdown of all RDT&E Costs by Year: RDT&E software costs and hardware costs are broken out by year using the input program management factors.

(10) Summary and Report Generation: The annual programming costs generated by module five and the production cost breakout by year, module 7, are summarized and a report is generated.

The following equation, referred to in this report as the NAVAIRDEVCEN software cost model, is basically module five, and provides an estimate of the total number of man-months required to develop a software package for an avionic computer system:

$$Y = 2.8X_2 + 1.3X_3 + 33X_4 - 17X_5 + 10X_6 + X_7 - 188 \qquad (1)$$

where Y = number of manmonths

X_2 = number of machine language instructions (thousands) in delivered program

X_3 = number of man-miles traveled by contractors

X_4 = number of document types produced

X_5 = average programmer's experience with system (NOTE: The experience "for the system programmer is the sum of the average number of years of experience with the specific computer-type, application, and language.")

X_6 = number of independent consoles

X_7 = percentage of new instructions

Module two can be used to calculate the variable X_2 based on the function of the program. An interesting comment found in a GRC report notes that the weighing factor applied to the documentation in this model is based upon pre-work standard (WS) 8506 experience. GRC, based upon a private communication, 31 Aug 73, states that the documentation costs can be expected to triple with the implementation of WS 8506. Hence, the X_4 term would be modified and the equation would appear as:

$$Y = 2.8X_2 + 1.3X_3 + 99X_4 - 17X_5 + 10X_6 + X_7 - 188 \quad (2)$$

7. AEROSPACE MODEL

The model referred to here as the "Aerospace Model" was taken from a 1975 Aerospace Corporation report on cost estimating (Reference 18). The data used to develop the cost equations for this model were divided into two groups or types of programming efforts, real-time programs, and support programs. Included in the cost data are costs that accrued as a result of problems encountered in developing a large-scale software program. The real-time software program development problem areas identified were:

a. Limited core storage of computers.

b. Timing requirements

c. Accuracy requirements.

d. Fixed-point arithmetic.

e. Changing specifications.

f. Real-time simulations.

1. Inability to interface languages.

2. Nonstandardization of computers between machines and operational program or support program problem areas identified were:

(a) Timing and accuracy problems.

(b) Inability to transfer simulation activities of one contractor to another due to language and machine differences.

(c) Inadequate and changing specifications.

(d) Lack of an organized method of defining endpoints and products of various development phases.

The data base used to develop the cost equation for real-time software program costs consisted of 13 large-scale programs, primarily airborne and space oriented programs. The cost equation derived from a regression analysis of those 13 data points. The cost equation developed is as follows:

$$\text{Man-months} = 0.057\ (\text{Instruction})^{0.94} \quad (3)$$

The sample size for operational support programs consisted of seven data points (both airborne and ground software programs were in the data base). The resulting equation for support software man-months estimation is:

$$\text{Man-Months} = 2.012\ (\text{Instruction})^{0.404} \quad (4)$$

The comment about language type mentioned above holds true in this case as well.

Once the number of man-months required for development is estimated using Equation 3 or 4, a dollar value per man-month is used to derive the total development cost. The estimated cost per man-month would obviously vary with the particular company performing the programming function. For planning purposes, an average of $5,000 per man-month is used by Aerospace Corp.

8. GENERAL RESEARCH CORPORATION MODEL

The GRC model was taken from a report entitled "Estimation of Computer Requirements and Software Development Costs", March 1974, prepared by M. A. Taback and M. C. Ditmore of General Research Corporation (Reference 12). The purpose of GRC was to determine a means of quantifying computer software development cost from overall system requirements. GRC had previously developed a procedure for determining the data processing speed and memory required to implement various computer functions from system performance requirements. The report presents a cost estimating relationship for computer software development which models the effects of the following: (1) program size, (2) computer language, (3) complexity, and (4) hardware constraints. The key conclusion of the report was that the program size, used along with the effects of program complexity, high-level language, and hardware constraints, is a reasonable predictor of software development cost.

The first step in utilizing any of the GRC models or any other such model, that of estimating the number of instructions for the particular program, appears to be the critical step. GRC suggests that one should develop the algorithms that are required and then utilize Table 6, which is a table of typical functional requirements in terms of number of instructions required, to implement the algorithm.

The CER developed by GRC used the factors which they felt could be identified either prior to program start up or immediately thereafter. The major factors include:

a. Estimated number of instructions.

b. Language used.

TABLE 6

FUNCTIONAL TRENDS IN AVIONICS MEMORY REQUIREMENTS

Typical Current Applications	Instructions and Constants*
Navigation	2600
Air-to-Air Weapon Delivery	930
Air-to-Ground Weapon Delivery	1120
Data Link	630
Tacan & Steering	120
Radar Update	45
Attitude Data	380
Displays and Control	1100
Self Test	570
Executive & Input/Output	1400
Common Subroutines	300

NADC Guidelines**	Instructions and Data
Radar Processing	16,000
Acoustical Processing	16,000
ASW Non-Acoustical Sensors	16,000
Navigation	8,000
Flight Control	8,000
Data Collection	8,000
Fire Control	8,000
Recon Data	8,000
Display Processing	16,000
Data Communications	8,000
Console and Cockpit	16,000

Projection for Near-Future Bomber	Instructions and Data
Navigation	17,000
Weapon Delivery	9,000
Target/Check Point Acquisition	4,500
Radar Homing, Location	14,500
Communications	10,000
Countermeasures	5,000
Mission Data Center	11,500
Controls, Display & Outputs	17,000
Miscellaneous	7,000

*Constants are fixed numbers that are used in computations and are prestored, along with the instructions that use them.

**Minimum requirements, main storage only (offline mass storage estimated separately).

c. Degree of difficulty or complexity.

d. Degree of saturation of the host computer, where degree of saturation refers to the amount of excess central processor speed and memory storage available to the programmer. The resulting CER provides the

$$C_s = 0.232\ N_i^{\ 1.43} \tag{5}$$

total software development cost in 1973 dollars assuming unlimited computer resources, where C_s = total software development cost and N_i = number of machine language instructions in the software development effort under consideration. Equation 5 does not include the extra cost incurred in the development of an operating system for a new computer or the modification of an existing operating system to accommodate the software program. Development cost for compilers, assemblers, and other support software must be handled as additional software to be developed.

Addressing software developments within the context of constrained computer resources, if we let P = fraction of maximum speed and memory capacity utilized the total constrained software cost, $(C_s)c$ is

$$(C_s)c = C_s \frac{0.7}{1 - \sqrt{P - 0.5}} \quad \text{for } P > 0.5 \tag{6}$$

If, for example, a system were to utilize 75 percent of its memory capacity (P = 0.75) then the CER reduces to:

$$(C_s)c = C_s \frac{0.7}{1 - \sqrt{0.25}} = 1.40\ C_s \tag{7}$$

GRC cited three primary effects of the use of a higher order language (HOL) and stressed the fact that these are a first approximation of the effect of an HOL:

1. One to three times as much storage space is required for a HOL as for a machine oriented language (MOL), depending on the type of language and the compiler used.

2. Execution of a HOL is one to three times slower than execution of MOL, depending on type of compiler.

3. Programming costs for a HOL are one-half to one-third those of MOL.

9. THE SYSTEM DEVELOPMENT CORPORATION MODEL

The System Development Corporation (SDC) in 1967 published a report entitled, "Management Handbook for the Estimation of Computer Programming Costs," based on work sponsored by the ESD (Reference 19). This report includes qualitative discussions/guidelines to help managers estimate costs of computer programming. SDC spent considerable time analyzing a large amount of data in an attempt to identify the dominant factors impacting programming costs. Table 7 presents the distribution of software programs in the SDC data base by programming application.

Through regression analysis on the 94 variables displayed in Table 8, SDC identified 12 variables which were sufficiently significant to use as estimating indices. For this analysis 105 programs categorized as the large computer subsample were selected from their software data base. The large computer subsample consisted of software developed for machines with a monthly rental price or equivalent purchase price of $750,000 or greater. Equation 8 estimates man-months per thousand instructions coded (Y) expressed in terms of the 12 variables identified.

$$Y = 0.049 + 15.2X_6 - 0.23X_{25} + 0.528X_{30} + 4.50X_{37} + 0.091X_{46} - 17.5X_{48.1} + 25.1X_{51} + 22.0X_{54} + 26.0X_{56} - 0.25X_{64} - 14.9X_{65} + 10.4X_{74} \quad (8)$$

where:

X_6 = Complexity of the program system interface. In the computer program, if more than 50 percent of the design effort is devoted to problems associated with transferring data to or from the program data point, $X_6 = 2$; if between 10 percent and 50 percent effort is devoted to data transfer problems, $X_6 = 1$; if less than 10 percent effort is devoted, $X_6 = 0$.

TABLE 7

DISTRIBUTION OF DATA BY PROGRAMMING APPLICATION (REFERENCE 19)

		TOTAL DATA POINTS	TYPE OF PROGRAM			
			BUSINESS	SCIENTIFIC	COMPUTER SOFTWARE	OTHER
GOVERNMENT	U.S. AIR FORCE*	38	26	10	2	
COMPUTER SOFTWARE RESEARCH AND DEVELOPMENT	Company A	6	3		3	
	Company B	1			1	
	Company C	1	1		0	
	Company D	69	17	12	5	35
COMPUTER HARDWARE AND AEROSPACE	Company E	2		2		
	Company F	3	2	1		
	Company G	21	19	2		
	Company H	28	11		17	
	TOTAL	169	79	27	28	35

*Data represent 14 separate USAF organizations.

TABLE 8

COMPUTER SOFTWARE VARIABLES (REFERENCE 19)

X_1	Vagueness of design requirements definition.
X_2	Innovation required.
X_3	Lack of knowledge of operational requirements.
X_4	Number of organizational users.
X_5	Number of ADP centers.
X_6	Complexity of program system interface.
X_7	Response time requirements.
X_8	Stability of design.
X_9	On-line requirements.
X_{10}	Total object instructions delivered.
X_{11}	Percent delivered object instructions reused.
X_{12}	Total nondelivered object instructions produced.
X_{13}	Total source instructions written.
X_{14}	Percent source instructions written in POL (Procedure Oriented Language).
X_{15}	Percent of total source instructions discarded.
X_{16}	Percent of total object instructions discarded.
X_{17}	Number of conditional branches.
X_{18}	Number of words in the data base.
X_{19}	Number of classes of items in the data base.
X_{20}	Number of input message types.
X_{21}	Number of output message types.
X_{22}	Number of input variables.
X_{23}	Number of output variables.

TABLE 8 (Cont'd)

X_{24}	Number of words in tables, and constants not in data base.
X_{25}	Percent clerical instructions.
X_{26}	Percent mathematical instructions.
X_{27}	Percent input/output instructions.
X_{28}	Percent logical control instructions.
X_{29}	Percent self-checking instructions.
X_{30}	Percent information storage and retrieval functions.
X_{31}	Percent data acquisition and display function.
X_{32}	Percent control or regulation function.
X_{33}	Percent decision-making functions.
X_{34}	Percent transformation functions.
X_{35}	Percent generation functions.
X_{36}	Average operating time.
X_{37}	Frequency of operation.
X_{38}	Insufficient memory.
X_{39}	Insufficient I/O capacity.
X_{40}	Stringent timing requirements.
X_{41}	Number of subprograms.
X_{42}	Programming language.
X_{43}	POL expansion ratio.
X_{44}	Support program availability.
X_{45}	Internal documentation.
X_{46}	External documentation.
X_{47}	Total number of document types.
X_{48}	Type of program (business, scientific, utility, other)

TABLE 8 (Cont'd)

X_{49}	Compiler or assembler used.
X_{50}	Developmental computer used.
X_{51}	First program on computer.
X_{52}	Average turn-around time.
X_{53}	ADP components developed concurrently.
X_{54}	Special display equipment.
X_{55}	Core capacity.
X_{56}	Random access device used.
X_{57}	Number of bits per word.
X_{58}	Memory access time.
X_{59}	Machine add time.
X_{60}	Compute cost.
X_{61}	Percent senior programmers.
X_{62}	Average programmer experience with language.
X_{63}	Average programmer experience with application.
X_{64}	Percent programmers participating in program design.
X_{65}	Personnel continuity.
X_{66}	Maximum number of programmers.
X_{67}	Lack of management procedures.
X_{68}	Number of agencies concurring in design.
X_{69}	Customer inexperience.
X_{70}	Computer operated by agency other than program developer.
X_{71}	Program developed at site other than the operational installation.
X_{72}	Different computers for programming and operation.

TABLE 8 (Concluded)

X_{73}	Closed or open shop operation.
X_{74}	Number of locations for program data point development.
X_{75}	Number of man trips.
X_{76}	Program data point developed by military organization.
X_{77}	Program data point developed on time-shared computer.
X_{78}	Complexity of system interface with other systems.
X_{79}	Security classification level.
X_{80}	Number of sources of system information.
X_{81}	Accessibility of system information.
X_{82}	Degree of system change expected during development.
X_{83}	Degree of system change expected during system operations.
X_{84}	Number of functions in the system.
X_{85}	Number of system components.
X_{86}	Number of components -- not off-the-shelf.
X_{87}	Percent senior analysts.
X_{88}	Quality of resource documents.
X_{89}	The availability of special tools.
X_{90}	Degree of standardization in policy and procedures.
X_{91}	Number of official reviews of documents.
X_{92}	Personnel turnover.
X_{93}	Output volume.
X_{94}	Input volume.

X_{25} = Percent of clerical instructions.

X_{30} = Percent of information storage and retrieval functions.

X_{37} = Frequency of operation. If this variable is not applicable, $X_{37} = 0$; if frequency of operation is less than one per month, $X_{37} = 1$; more than one per month and less than one per week, $X_{37} = 2$; more than one per week and less than one per day, $X_{37} = 3$; if daily, $X_{37} = 4$; if utility or on-line, (including compilers) $X_{37} = 5$.

X_{46} = External documentation. This is the number of pages written for, or distributed to, customers.

$X_{48.1}$ = Business. For programs classified as business application, $X_{48.1} = 1$; the remaining applications, $X_{48.1} = 0$.

X_{51} = First program on computer. If it is a new machine or new to the installation and to the programmers, $X_{51} = 1$. If old or not new, $X_{51} = 0$.

X_{54} = Special display equipment involving use of graphic displays, CRTs, scopes, etc. $X_{54} = 1$ if used, $X_{54} = 0$ if not used.

X_{56} = Random access device used such as drum, disc, etc. $X_{56} = 1$ if used, $X_{56} = 0$ if not used.

X_{64} = Percent programmers participating in program design. This is the ratio of programmers participating in the design of the program to the total number of programmers assigned to the program development.

X_{65} = Personnel continuity, specifically, the number of personnel working for the duration of the project divided by the maximum number assigned at any time.

X_{74} = Number of locations for program development.

As GRC points out in Reference 12, this equation requires rather detailed previous knowledge of software parameters, and when such information is unavailable or cannot be estimated with accuracy, the technique cannot be used.

SECTION VI

APPLICATION OF MODELS

All but one of the models presented in Section V of this report were used to analyze cost associated with six large scale computer programs. Because of the lack of sufficient data, the SDC model was not applicable. Actual expended costs were known on two of the six software packages analyzed. Although this exercise should not be considered as a validation or even an evaluation of the models, the results may assist software managers in becoming sensitive and aware of what factors affect such engineering economic estimates.

The six programs will be described in as much detail as was needed to apply the CER's and the basic assumptions that were made will be presented. Input data requirements for the eight models under consideration are presented in Table 9. Table 10 displays input data assumed for this exercise for each of the software development efforts analyzed.

1. DIGITAL AVIONICS INFORMATION SYSTEM (DAIS)

The Digital Avionics Information System (DAIS) Air Superiority Program contained 36,916 statements and was programmed in a higher order language, JOVIAL (J73/I). The program was broken down into nine modules (executive, navigation, weapon delivery, ECM, control/display, flight control, management, communications, DAIS Integrated Test System) by number of instructions and type of program, i.e., real-time, operating system, utility, or application. The following facts and assumptions were made pertaining to each module: (1) executive is an operating system containing 4000 instructions, (2) navigation containing 3848 instructions was real-time program application type, (3) weapon delivery is a real-time program application type containing 3272 instructions, (4) ECM contained 534 instructions and is a real-time application type program, (5) control/display containing 3663 instructions is a real-time utility program, (6) flight control, real-time application program includes 8770 instructions, (7) management containing 7838

TABLE 9

SOFTWARE COST ESTIMATE INPUT WORKSHEET, DEFINITION OF TERMS, AND SOFTWARE OUTPUT SHEET

LANGUAGE TYPE _____ (HOL or MOL)

If HOL, give name _____

ESTIMATE NUMBER OF TOTAL INSTRUCTIONS _____

BREAKOUT OF TYPE OF INSTRUCTIONS:

# of control instructions _____
pre-post CPU instructions _____
algorithm instructions _____
data management instructions _____
real time instructions _____

DELIVERED OR OPERATING INSTRUCTIONS? _____

OBJECT OR SOURCE INSTRUCTIONS? _____

NEW OR OLD PROGRAM (% new, % old)? _____

RANK DIFFICULTY (between 1 and 9, 1 = easy, 9 = hard) _____

MANRATED OR NON-MANRATED? _____

IF UNFAMILIAR PROGRAM (NEW), RANK BETWEEN 1.5 and 2.0 _____
(Subjective ranking of how unfamiliar program is)

OPERATING SYSTEM OR NOT? _____

REAL TIME SYSTEM OR NOT (%)? _____

MAN MILES TRAVELED BY CONTRACTOR _____

DOC TYPES _____

SYSTEM PROGRAMMER EXPERIENCE (YEARS) _____

INDEPENDENT CONSOLES _____

SUBROUTINES _____

REUSEABLE SUBROUTINES _____

FUNCTION OF PROGRAM (BRIEF STATEMENT) _____

TABLE 9 (Cont'd)

LANGUAGE - Self explanatory

NUMBER OF INSTRUCTIONS - Self explanatory

CONTROL INSTRUCTION - Controls execution flow and is non-time critical.

PRE-POST CPU INSTRUCTION - Pre- or post algorithm processor which manipulates data for subsequent processing or output.

ALGORITHM INSTRUCTION - Which performs logical or mathematical operations

DATA MANAGEMENT INSTRUCTIONS - Data management routine which manages data transfer within the computer.

REAL-TIME INSTRUCTIONS - Time critical processor which is highly optimized machine dependent code.

DELIVERED OR OPERATING INSTRUCTIONS - Delivered is the total of instructions received from contractor as opposed to the actual instructions you would use (operating). Contractor may have to simulate your machine (system).

OBJECT VS SOURCE INSTRUCTIONS - Object is machine language instructions after source deck has been compiled. Source → compiler → object

NEW OR OLD - Self explanatory

RANK DIFFICULTY - Subjective 1=easy 5=median 9=difficult

MANRATED - NON-MANRATED - Self explanatory

UNFAMILIAR - For unfamiliar, multiply by 1.5 - 2.0. A judgement of how unfamiliar the program is to the programmer.

OPERATING SYSTEM - Software which controls the execution of computer programs and which may provide scheduling, debugging, I/O control, accounting, compilation, storage assignment, data management, and related services. Operating system program component, of a system, costs more per instruction, than the application or utility program components.

REAL TIME - Real time programs are those in which the time is kept as a variable, stored in memory, to be incremented or stepped under program control. It is used to describe processes in which the computer is controlling a device and must receive input signals and transmit output signals within the certain maximum time. For example, SAT. control, ship control, flight control, navigation.

MAN-MILES TRAVELED - Miles per man traveled by the contractor to and from the customer.

DOC TYPES - Reports, flow charts, user manuals, etc.

SYSTEM PROGRAMMER EXP. - Total years of experience with the particular system.

SUBROUTINES - Self explantory

OF REUSED SUBROUTINES - Reused from previous programs.

TABLE 9 (Concluded)

SOFTWARE OUTPUT SHEET

_____

(Name of Software Program)

TOTAL COST TO DEVELOP PROGRAM ($75) _____

COST OF ANALYSIS $_____
DESIGN $_____
CODE $_____
TEST $_____
DOCUMENTATION $_____

TOTAL MAN MTHS (@ $3930/MAN MTH) _____

TABLE 10

INPUT DATA

NAME OF PROGRAM	NUMBER OF INSTR.	HOL	MOL	OPERATING SYSTEM	REAL TIME APPLICATION	REAL TIME UTILITY	APPLICATION	UTILITY	LEARNING CURVE	DIFFICULTY
I. DAIS Air Superiority	36,916	X JOVIAL							1.7	
Executive	4,000			X						
Navigation	3,848				X					
Weapon Delivery	3,272				X					
ECM	534				X					
Control/Display	3,663					X				
Flt Control	8,770				X					
Management	7,838			X						
Communication	9,991				X					
DITS	4,000							X		
II. DAIS Close Air Support	37,169	X JOVIAL							1.7	
Executive	8,000			X						
Navigation	5,939				X					
Weapon Delivery	6,534				X					
ECM	534				X					
Control/Display	3,663					X				
Flt Control	496				X					
Management	7,838			X						
Communication	991				X					
Subroutines	3,174							X		
III. J73/I Compiler	132,087	X						X	8.5	
IV. SDVS	82,965	X		X				X	9.0	

TABLE 10 (Concluded)

NAME OF PROGRAM	NUMBER OF INSTR.	HOL	MOL	OPERATING SYSTEM	REAL TIME UTILITY/APPLICATION	WOLVERTON CATEGORY	LEARNING CURVE	DIFFICULTY
V. JTIDS Mini		X FORTRAN OR JOVIAL						
Executive	1,000			X		T		
Math Sub	876					OM/A		
Self Test	1,271					NM/P		
Input Buffer	25					NM/P		
Message Filter	180					NM/P		
Data Base	7,020			X		NM/P		
Prompter	120				X	NM/P		
Situation Display	1,340				X	NM/P		
Received Comm. Processing	3,620				X	NM/P		
Ackn. Function	60				X	NM/P		
Message Const.	466					NM/P		
Parameter Storg.	450					NM/P		
VI. JTIDS Micro		X FORTRAN OR JOVIAL						
Executive	1,450			X		T		
Math Sub	1,270					OM/A		
Self Test	1,719					NM/P		
Input Buffer	36					NM/P		
Message Filter	261					NM/P		
Data Base	7,155			X		NM/P		
Prompter	174				X	NM/P		
Situation Display	1,745				X	NM/P		
Received Comm. Processing	4,349				X	NM/P		
Ackn. Function	83				X	NM/P		
Message Const.	579					NM/P		
Parameter Storg.	450					NM/P		

instruction is an operating system, (8) communications containing 9991 instructions is a real-time application, and (9) DITS is a utility program of 4000 instructions. The total DAIS air superiority program was man-rated.

2. DAIS CLOSE AIR SUPPORT

The DAIS Close Air Support program contained 37,169 statements. It was also programmed in JOVIAL (J73/I) and broken down into nine modules (Executive, Navigation, Weapon Delivery, ECM, Control/Display, Flight Control, Management, Communications, Subroutines) by number of instructions and type of program, i.e., real-time, operating system, utility, or application. The following assumptions were made pertaining to each module: (1) executive containing 8000 instructions was an operating system, (2) the navigation, weapon delivery, ECM, flight control, and communication modules are all real-time application programs with 5939, 6534, 534, 496, and 991 instructions respectively, (3) the Control/Display module is a utility real-time program of 3663 instruction, (4) management containing 7838 instruction is an operating system, and (5) Subroutines contain 3174 instructions and was a utility program.

For both the DAIS Air Superiority and Close Air Support programs, a subjective "point on the learning curve" of 1.7 was assumed. This number is based on the ESD model (Table 4) where "if unfamiliar, multiply by 1.5 to 2.0." (Reference 8)

3. F-15 JOINT TACTICAL INFORMATION DISTRIBUTION SYSTEM (JTIDS)

The F-15 Joint Tactical Information Distribution System (JTIDS) software program has two versions, one for minicomputers, one for microcomputers. The information about these programs was gathered from Reference 15. Both the mini- and micro-computer JTIDS software programs consisted of 12 modules. Table 11 displays the basic information about each module. More detailed information is contained in the reference.

TABLE 11

JTIDS PROGRAM BREAKDOWN

Name	Mini Instructions	Wolverton Cost Category	Instructions	Operating System or Real-Time
Executive	1000	T	1450	Operating
Mate Subroutine	876	OM/A	1270	Neither
Self Test	1271	NM/P	1719	Neither
Input Message Buffer	25	NM/P	36	Neither
Message Type Filter	180	NM/P	261	Neither
Data Base Management	7020	NM/P	7155	Operating Sys
Prompter	120	NM/P	174	Real-time
Situation Display	1340	NM/P	1745	Real-time
Received Command Processing	3620	NM/P	4349	Real-time
Acknowledge Function	60	NM/P	83	Real-time
Message Construction	466	NM/P	579	Neither
Parameter Storage	450	NM/P	450	Neither

Both mini- and micro-computer JTIDS programs utilized the higher ordered languages of FORTRAN or JOVIAL.

4. DAIS SUPPORT SOFTWARE

The last two programs to be analyzed were the DAIS J73/I compiler and the Software Design and Verification System (SDVS). Both the J73/I compiler and the SDVS programs are good data points since we have actual costs to compare the estimates against.

The J73/I compiler consisted of 132,087 source instructions, written in HOL. It was an old program which was modified, and ranked 8.5 in difficulty (1 = easy, 10 = hard). The program was a man-rated, nonoperating system, and non-real-time program. Total man-miles traveled by contractor was 68,400. The number of document types was five, with three on-line, remote terminals, and 50 percent of the program being new instructions. The average of the programmers' experience was assumed to be three years.

The Software Design and Verification System (SDVS) was also written in J73/I and had 82,965 source instructions. SDVS was a non-man-rated, operating system program with 9.0 degree difficulty. The program was non-real-time with eight document types. An average of 21,600 miles was traveled by contractors with an average of 2.8 years software experience. The number of on-line remote terminals is five with 100 percent new instructions.

5. RESULTS

Table 12 is a summary chart of results obtained by applying the eight software cost estimating models. For application of those models, which were not formulated in terms of the RCA PRICE model, economic inflation rates were utilized to adjust the results to 1977 dollars. Those cases where the information obtained was insufficient to utilize a particular methodology are identified on Table 12.

TABLE 12

RESULTS OF COST MODELS

Estimates / Programs	WOLVERTON	MOD WOLVERTON	ESD	TECOLOTE	NAVAIRDEVCEN	AEROSPACE
DAIS AIR SUPERIORITY	1,695,841	1,509,493	2,126,470	1,406,479	Insufficient Data	2,990,730
DAIS CLOSE AIR SUPERIORITY	2,237,306	2,243,254	2,043,376	1,417,859	Insufficient Data	2,681,614
F-15 JTIDS MINI	834,196	855,751	749,975	376,441	Insufficient Data	1,032,545
F-15 JTIDS MICRO	972,148	997,709	881,012	458,831	Insufficient Data	1,203,514
J73/I COMPILER	Insufficient Data	Insufficient Data	792,522	3,181,375	1,826,507	926,505
SDVS	Insufficient Data	Insufficient Data	1,866,713	1,837,791	1,744,448	767,811

TABLE 12 (Concluded)

	IBM	GRC	AVERAGE	ACTUAL
DAIS AIR SUPERIORITY	536,681	3,792,174	2,008,266 $54/Instruc.	
DAIS CLOSE AIR SUPERIORITY	518,183	3,829,393	2,138,726 $57/Instruc.	
F-15 JTIDS MINI	189,312	1,191,399	747,088 $45/Instruc.	*NOTE
F-15 JTIDS MICRO	228,482	1,496,871	891,224 $46/Instruc.	*NOTE
J73/I COMPILER	1,862,427	23,474,766	5,344,017 $40/Instruc.	642,039 $5/Instruc.
SDVS	1,096,824	12,072,330	3,220,986 $38/Instruc.	1,103,000 $13/Instruc.

*NOTE: McDonnell Douglas estimated between $4.4M to $4.7M for F-15 JTIDS MINI and $5.1M to $5.5M for F-15 MICRO.

SECTION VII

SUPPORT/MAINTENANCE COST AREA

Once the software program has been accepted, continual support must be furnished to modify the software package to meet changing mission and performance requirements. Besides the modifications to software programs, corrections must be made to previously undetected errors which occur. Because software is the controlling and integrating agent in weapon systems today, proper support is required to insure that the program performs its intended functions properly. In avionic software programs this support is critical since errors could result in inadvertent armament release or impact with the ground in terrain following modes.

The software development process is typically oriented toward minimizing the total development time or maximizing the program's efficiency. In a study on the relative amount of time spent on software maintenance it was shown that most software facilities spent somewhere between 20 and 30 percent of their time on software maintenance, but some installations spent 90 to 100 percent of their time maintaining software. Air Force avionics software is much like the latter and "currently it costs something like $75 per instruction to develop the software, but the maintenance of the software has cost up to $4,000 per instruction." (Reference 20). Further noted by Judith A. Clapp (The MITRE Corp) in "A Review of Software Cost Estimation Methods" (Reference 18) was the fact that 54 percent of all errors were found after acceptance tests were conducted and of these 84 percent were design errors; also, of the total number errors found, 64 percent were attributed to mistakes in design. Throughout the development phase relatively little thought is usually given about what will happen after development is completed. According to the CCIP-85 Report (Reference 5), three things are likely to happen after development:

(1) Another organization will want to use all or part of the software for its application, (2) the user will upgrade eventually to a new machine and will wish to convert the software, and (3) users will quite frequently want the programs changed to meet new requirements, produce new reports, accommodate new inputs, clear up inconsistencies, add new options, etc.

Software transferability involves addressing the ease with which the first two points mentioned above can be accomplished. Maintainability, quite simply stated, involves the capabilities to satisfy the last point. Both the transferability and maintainability aspects involve considerable costs and inconveniences. A couple of prime examples of the costs involved with transferability and maintainability were given in the CCIP-85 Report: Strategic Air Command (SAC) estimated it would take three years for 200 programmers to convert the SAC Control System (SACCS) software to the upcoming SAC Worldwide Military Command and Control System computer. This is equivalent to three years worth of delays and roughly $30 million in costs. It took 150 programmers one year to convert software for Electronic Intelligence (ELINT) and Minuteman application onto the IBM 360/85; currently the 360/85 has about 75 maintenance programmers. The PACER software cost $8 million to develop and is maintained by about 50 programmers, which is an annual software maintenance cost of about 25 percent of development costs. Conversion and maintenance expenses could be reduced by such things as: machine-independent, problem-oriented programming languages; use of structured programming techniques; development of computer software maintenance and transfer aids; maintenance of an Air Force software library; and formulation of a standard for computer hardware, software, terminology and documentation. "Reduction in proliferation of different computer hardware and software styles would reduce the high cost of retraining, particularly considering Air Force officer rotation policies: The Keesler Training Center spent $9.6 million in training computer analysts, programmers, operators, and maintenance personnel in FY69." (Reference 5)

A search of current literature results in very little in the way of predicting support and maintenance cost for computer software. Unlike hardware operation and support (O&S) models, where the cost of spares, maintenance manhours, materials, training, etc., can be estimated based on some physical characteristics of the system, software maintenance is strictly a function of manhours to perform the necessary actions. Thus far, maintenance costs for software seem to be primarily an engineering estimate by an expert, someone familiar with the changes to be made to

a program, rather than putting certain parameters into an CER or formula and calculating annual maintenance costs. The effects of the structured programming or chief programmer approaches on maintenance costs can only be subjectively estimated as of this time.

The "Aerospace Model", was discussed in Section V-7, is a total life cycle cost model. The procedure permits costs for design and development (D&D), investment, and operations and maintenance (O&M) to be determined in a series of prearranged steps. "The model first calculates hardware (CPU) costs, then applies factors for estimating the other D&D, investment, and O&M costs, and finally summarizes the total program costs." (Reference 21). Most of the factors in the model were developed based on a report entitled, "Investment Costs for Flight Area Defense Systems," also referred to as the FADS study. The primary maintenance equations for software appear as follows:

Software training costs during production phase:

Initial Civilian = number of men X 27,200

Initial Contractor = number of men X 35,598

Initial Military = number of men X 17,400

b. During the deployment phase:

Personnel contractor support cost = (number of men) X $48,000 X (number of years O&M or deployment)

Military support cost = (number of men) X $18,000 X (number of years O&M or deployment)

These equations should only be used if the estimator has no prior basis for determining costs of any of his data-processing system elements. The model, as mentioned above, calculates hardware and software costs, and is referred to as a "Data Processing System Cost Model".

In the past, software support has been performed by the contractor long after transition to the user. Because of the nature of most airborne software systems, the contractor alone had the expertise and equipment to perform these follow on maintenance actions and modification. "Large operational flight programs (OFP's) usually are never absolutely debugged, and errors can remain undetected for long periods of time due to the extremely large number of logic paths." (Reference 5). Interesting to note at this time is that currently the largest program which has been mathematically proven correct has about 400 instructions; on the other hand, experience with the SAC Control System (SACCS) program, with about 2.7 million instructions, indicates that about one software error per day is discovered (Reference 5). Figure 3 summarizes current experience in software meantime between failure (MTBF) in days as a function of program size in instructions.

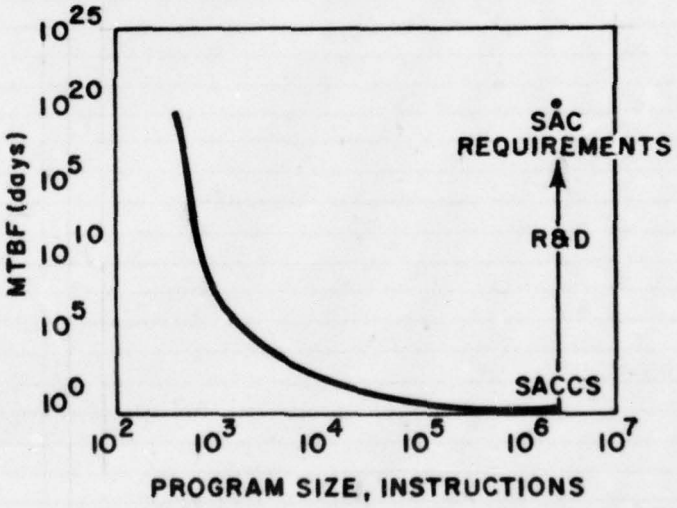


Figure 3. Program Size vs Reliability

The Air Force has recognized the dependence upon contractors for maintenance support and has taken steps to develop in-house capabilities to support current and future OFP's. Two recent studies of software management, F-111 and A-10, show promising results of Air Force in-house capabilities.

To combat the maintenance costs in 1974 work began on an Avionics Integration and Support Facility (AISF) at the Sacramento Air Logistics Center (ALC). All ALC's software programs are to be supported by AISF. The AISF facility will provide hardware/software integration support as well as a dynamic simulation facility. The purpose of the dynamic simulator is to execute the OFP's in a bit by bit fashion to debug the software functions.

The large computer at AISF will make it easier, safer, and less expensive than flight tests to validate OFP's. Not only will the AISF facility support OFP data processing, technical data, and procedure verification, but also provide air crew familiarization and training. The AISF is costing approximately $20 million for development and implementation. There are presently 40 contractors amd 60 Air Force military and civilian engineers working at the AISF facility. Estimated annual recurring costs for the facility are about $3 million (Reference 15).

Since there is a lack of fully validated O&M predictive models for software programs, (Aerospace, and the forthcoming GRC "LCC software model" are the only "existing" models at the present time), this AISF facility can hopefully provide a point of contact to supply O&M cost estimates for software programs under development.

APPENDIX A

This appendix contains the computer program listing of the Modified Wolverton model. The next two pages contain the listing with the remaining pages containing an example output listing. The following inputs are required:

NC = Number of control routine instructions

NIO = Number of input/output instructions

NP = Number of Pre- or Post- algorithm processor instructions

NA = Number of algorithm instructions

ND = Number of data management instructions

NT = Number of time critical processor instructions

The above six inputs were defined in Section V-1, and utilized the following format statement: 12 FORMAT(615)

For the example the following values were used:

NC=0 NIO=0 NP=4964 NA=5702 ND=7155 NT=1450

```
      24  FORMAT(   5X,3H = ,F10.2,23H ALGORITHM              ,5X,11H TEST
     1   ,F12.2,6X,I5,15X,F10.2)
   60  WRITE(6,25)CDO,DUC,ND,UD
      25  FORMAT(   5X,3H = ,F10.2,23H DATA MANAGEMENT        ,5X,11H DOCUMEN
     1T =,F12.2,6X,I5,15X,F10.2)
       WRITE(6,28)CTO,NI,UT
       WRITE(6,36)
   65  36   FORMAT(8X,10H==========,41X,10H==========)
       WRITE(6,27)TCO,COST
      27  FORMAT(   5X,3H = ,F10.2,23H TOTAL  BY  INSTRUCTION,5X,11H TOTAL
     1S  ,F12.2)
       WRITE(6,41)COST,NI
   70  41  FORMAT(///, 7H COST =,F12.2, 5H FOR ,I8, 14H INSTRUCTIONS )
       CCN=35.57143-.23333333333*D+.0052380952*(D**2)
       CIN=23.21429+.200000   *D+ .00071428571*(D**2)
       CPN=25.071429+.125     *D+ .0010714286 *(D**2)
       CAN=23.00   +    .08   *D+.001         *(D**2)
   75  CDN=31.25   +.304166666*D+ .00041666666*(D**2)
      28  FORMAT(   5X,3H = ,F10.2,23H REAL TIME              ,34X,I5,15X,F10
     1.2)
       LC=CCN
       LI=CIN
   80  LP=CPN
       LA=CAN
       LD=CDN
       LT=75.
       CCN=RNC*CCN
   85  CIN=RNI*CIN
       CPN=RNP*CPN
       CAN=RNA*CAN
       CDN=RND*CDN
       CTN=75.*RNT
   90  TCN=CCN+CIN+CPN+CAN+CDN+CTN
       TCO=1.27*TCN
       COST=TCN
       ANL=.200*COST
       DES=.187*COST
   95  COD=.217*COST
       TES=.283*COST
       DOC=.113*COST
       WRITE(6,31)CCN,ANL,NC,UC
      31  FORMAT(//,5X,3H = ,F10.2,15H CONTROL   NEW ,13X,11H ANALYSIS =,
  100 1F12.2,6X,I5,15X,F10.2)
       WRITE(6,22)CIN,DES,NIO,UI
       WRITE(6,23)CPN,COD,NP,UP
       WRITE(6,24)CAN,TES,NA,UA
       WRITE(6,25)CDN,DOC,ND,UD
  105  WRITE(6,28)CTN,NI,UT
       WRITE(6,36)
       WRITE(6,27)TCN,COST
       WRITE(6,41)COST,NI
      40 CONTINUE
  110  END
```

Percent → %

		10.% DIFICULTY	TME EFFORT		INSTRUCTIONS	UNIT COST
$	0.00	CONTROL OLD	ANALYSIS $	103484.72	0	18.34
$	0.00	INPUT/OUTPUT	DESIGN	96758.22	0	14.60
$	80653.21	PRE POST CPU	CODE	112280.92	4964	16.25
$	72225.32	ALGORITHM	TEST	146430.88	5702	12.67
$	145791.64	DATA MANAGEMENT	DOCUMENT $	58468.87	7155	20.38
$	108750.00	REAL TIME			1450	75.00
	==========			==========		
$	517423.62	TOTAL BY INSTRUCTION	TOTAL $	517423.62		

COST $ 517423.62 FOR 19271 INSTRUCTIONS

$	0.00	CONTROL NEW	ANALYSIS $	157956.05	0	33.76
$	0.00	INPUT/OUTPUT	DESIGN	147688.91	0	25.29
$	131191.43	PRE POST CPU	CODE	171382.32	4964	26.43
$	136277.80	ALGORITHM	TEST	223507.82	5702	23.90
$	245655.00	DATA MANAGEMENT	DOCUMENT $	89245.17	7155	34.33
$	108750.00	REAL TIME			1450	75.00
	==========			==========		
$	789780.27	TOTAL BY INSTRUCTION	TOTAL $	789780.27		

COST $ 789780.27 FOR 19271 INSTRUCTIONS

SOFTWARE CER

COST

20.% DIFICULTY

		THE EFFORT		INSTRUCTIONS	UNIT COST
$	0.00 CONTROL OLD	ANALYSIS $	113944.21	0	21.25
$	0.00 INPT/OUTPUT	DESIGN	106537.84	0	17.50
$	84388.03 PRE POST CPU	CODE	123629.47	4964	17.00
$	85629.98 ALGORITHM	TEST	161231.06	5702	15.00
$	169931.25 DATA MANAGEMENT	DOCUMENT $	64378.48	7155	23.75
$	108750.00 REAL TIME			1450	75.00
	==========		==========		
$	569721.05 TOTAL BY INSTRUCTION	TOTAL $	569721.05		

COST $ 569721.05 FOR 19271 INSTRUCTIONS

$	0.00 CONTROL NEW	ANALYSIS $	167285.54	0	33.00
$	0.00 INPT/OUTPUT	DESIGN	156411.98	0	27.50
$	138992.00 PRE POST CPU	CODE	181504.81	4964	28.00
$	142550.00 ALGORITHM	TEST	236709.04	5702	25.00
$	268312.50 DATA MANAGEMENT	DOCUMENT $	94516.33	7155	37.50
$	108750.00 REAL TIME			1450	75.00
	==========		==========		
$	836427.72 TOTAL BY INSTRUCTION	TOTAL $	836427.72		

COST $ 836427.72 FOR 19271 INSTRUCTIONS

SOFTWARE CEM

30.% DIFICULTY

		THE EFFORT		INSTRUCTIONS	UNIT COST
$ 0.00	CONTROL OLD	ANALYSIS $	123162.80	0	23.76
$ 0.00	INPUT/OUTPUT	DESIGN	115157.22	0	20.04
$ 88323.77	PRE POST CPU	CODE	133631.64	4964	17.79
$ 96933.98	ALGORITHM	TEST	174275.37	5702	17.00
$ 190885.18	DATA MANAGEMENT	DOCUMENT $	69586.98	7155	26.68
$ 108750.00	REAL TIME			1450	75.00
$ 615814.02	TOTAL BY INSTRUCTION	TOTAL $	615814.02		

COST $ 615814.02 FOR 19271 INSTRUCTIONS

$ 0.00	CONTROL NEW	ANALYSIS $	177326.32	0	33.29
$ 0.00	INPUT/OUTPUT	DESIGN	165800.11	0	29.86
$ 147856.29	PRE POST CPU	CODE	192399.06	4964	29.79
$ 149962.60	ALGORITHM	TEST	250916.75	5702	26.30
$ 291566.25	DATA MANAGEMENT	DOCUMENT $	100189.37	7155	40.75
$ 108750.00	REAL TIME			1450	75.00
$ 886631.62	TOTAL BY INSTRUCTION	TOTAL $	886631.62		

COST $ 886631.62 FOR 19271 INSTRUCTIONS

	40.% DIFICULTY	THE EFFORT		INSTRUCTIONS	UNIT COST
$ 0.00	CONTROL OLD	ANALYSIS $	131140.50	0	25.88
$ 0.00	INPUT/OUTPUT	DESIGN	122616.37	0	22.20
$ 92460.44	PRE POST CPU	CODE	142287.44	4964	18.63
$ 106437.32	ALGORITHM	TEST	185563.81	5702	18.67
$ 208453.43	DATA MANAGEMENT	DOCUMENT $	74094.38	7155	29.16
$ 108750.00	REAL TIME			1450	75.00
$ 655702.50	TOTAL BY INSTRUCTION	TOTAL $	655702.50		

COST $ 655702.50 FOR 19271 INSTRUCTIONS

$ 0.00	CONTROL NEW	ANALYSIS $	188078.40	0	34.62
$ 0.00	INPUT/OUTPUT	DESIGN	175853.30	0	32.36
$ 157784.39	PRE POST CPU	CODE	204065.06	4964	31.79
$ 158515.40	ALGORITHM	TEST	266130.93	5702	27.80
$ 315416.29	DATA MANAGEMENT	DOCUMENT $	106264.30	7155	44.08
$ 108750.00	REAL TIME			1450	75.00
$ 940392.00	TOTAL BY INSTRUCTION	TOTAL $	940392.00		

COST $ 940392.00 FOR 19271 INSTRUCTIONS

	50.% DIFICULTY		TIME EFFORT		INSTRUCTIONS	UNIT COST
$	0.00	CONTROL OLD	ANALYSIS $	137877.30	0	27.60
$	0.00	INPUT/OUTPUT	DESIGN	128915.26	0	24.00
$	96798.03	PRE POST CPU	CODE	149596.87	4964	19.50
$	114040.00	ALGORITHM	TEST	195096.38	5702	20.00
$	223236.00	DATA MANAGEMENT	DOCUMENT $	77900.68	7155	31.20
$	108750.00	REAL TIME			1450	75.00
$	689386.51	TOTAL BY INSTRUCTION	TOTAL $	689386.51		

COST $ 689386.51 FOR 19271 INSTRUCTIONS

$	0.00	CONTROL NEW	ANALYSIS $	199541.77	0	37.00
$	0.00	INPUT/OUTPUT	DESIGN	186571.55	0	35.00
$	168776.00	PRE POST CPU	CODE	216502.82	4964	34.00
$	168209.00	ALGORITHM	TEST	282351.60	5702	29.50
$	339862.50	DATA MANAGEMENT	DOCUMENT $	112741.10	7155	47.50
$	108750.00	REAL TIME			1450	75.00
$	997708.83	TOTAL BY INSTRUCTION	TOTAL $	997708.83		

COST $ 997708.83 FOR 19271 INSTRUCTIONS

SOFTWARE CER

60.% DIFICULTY THE EFFORT INSTRUCTIONS UNIT COST

$	0.00	CONTROL OLD	ANALYSIS $ 143373.21	0	28.92
$	0.00	INPUT/OUTPUT	DESIGN 134053.95	0	25.43
$	101336.54	PRE FCST CPU	CODE 155559.93	4964	20.41
$	119742.02	ALGORITHM	TEST 202873.09	5702	21.00
$	234632.89	DATA MANAGEMENT	DOCUMENT $ 81005.86	7155	32.79
$	108750.00	REAL TIME		1450	75.00
	==========		==========		
$	716866.05	TOTAL BY INSTRUCTION	TOTAL $ 716866.05		

COST $ 716866.05 FOR 19271 INSTRUCTIONS

$	0.00	CONTROL NEW	ANALYSIS $ 211716.42	0	40.43
$	0.00	INPUT/OUTPUT	DESIGN 197954.86	0	37.79
$	190831.43	PRE FCST CPU	CODE 229712.32	4964	36.43
$	179042.80	ALGORITHM	TEST 299578.74	5702	31.40
$	364005.00	DATA MANAGEMENT	DOCUMENT $ 119619.78	7155	51.00
$	104750.00	REAL TIME		1450	75.00
	==========		==========		
$	1058582.12	TOTAL BY INSTRUCTION	TOTAL $ 1058582.12		

COST $ 1058582.12 FOR 19271 INSTRUCTIONS

SOFTWARE CEM

	70.% DIFICULTY		THE EFFORT		INSTRUCTIONS	UNIT COST
$	0.00	CONTROL OLD	ANALYSIS $	147628.22	0	29.85
$	0.00	INPUT/OUTPUT	DESIGN	138032.39	0	26.49
$	106075.98	PRE POST CPU	CODE	160176.62	4964	21.37
$	123543.39	ALGORITHM	TEST	208893.93	5702	21.67
$	242844.10	DATA MANAGEMENT	DOCUMENT $	83409.95	7155	33.94
$	108750.00	REAL TIME			1450	75.00
	==========			==========		
$	738141.11	TOTAL BY INSTRUCTION	TOTAL $	738141.11		

COST $ 738141.11 FOR 19271 INSTRUCTIONS

$	0.00	CONTROL NEW	ANALYSIS $	224602.38	0	44.90
$	0.00	INPUT/OUTPUT	DESIGN	210003.22	0	40.71
$	193950.57	PRE POST CPU	CODE	243693.58	4964	39.07
$	191017.00	ALGORITHM	TEST	317812.36	5702	33.50
$	390543.75	DATA MANAGEMENT	DOCUMENT $	126900.34	7155	54.58
$	108750.00	REAL TIME			1450	75.00
	==========			==========		
$	1123011.88	TOTAL BY INSTRUCTION	TOTAL $	1123011.88		

COST $ 1123011.88 FOR 19271 INSTRUCTIONS

SOFTWARE CEH

	80.% DIFICULTY		THE EFFORT		INSTRUCTIONS	UNIT COST
$	0.00	CONTROL OLD	ANALYSIS $	150642.34	0	30.37
$	0.00	INPUT/OUTPUT	DESIGN	140850.59	0	27.18
$	111016.34	PRE POST CPU	CODE	163446.94	4964	22.36
$	125444.09	ALGORITHM	TEST	213158.91	5702	22.00
$	247869.64	DATA MANAGEMENT	DOCUMENT $	85112.92	7155	34.64
$	108750.00	REAL TIME			1450	75.00
	==========			==========		
$	753211.69	TOTAL BY INSTRUCTION	TOTAL $	753211.69		

ST $ 753211.69 FOR 19271 INSTRUCTIONS

$	0.00	CONTROL NEW	ANALYSIS $	238199.62	0	50.43
$	0.00	INPUT/OUTPUT	DESIGN	222716.65	0	43.79
$	208133.43	PRE POST CPU	CODE	258446.59	4964	41.93
$	204131.60	ALGORITHM	TEST	337052.46	5702	35.80
$	416778.75	DATA MANAGEMENT	DOCUMENT $	134582.79	7155	58.25
$	108750.00	REAL TIME			1450	75.00
	==========			==========		
$	1190998.10	TOTAL BY INSTRUCTION	TOTAL $	1190998.10		

ST $ 1190998.10 FOR 19271 INSTRUCTIONS

SOFTWARE CER

	$0.% DIFICULTY		THE EFFORT		INSTRUCTIONS	UNIT COST
$	0.00	CONTROL OLD	ANALYSIS $	152415.56	0	30.50
$	0.00	INPUT/OUTPUT	DESIGN	142508.55	0	27.50
$	116157.63	PRE POST CPU	CODE	165370.88	4964	23.40
$	125444.14	ALGORITHM	TEST	215668.02	5702	22.00
$	249709.50	DATA MANAGEMENT	DOCUMENT $	86114.79	7155	34.90
$	108750.00	REAL TIME			1450	75.00
$	762077.80	TOTAL BY INSTRUCTION	TOTAL $	762077.80		

COST $ 762077.80 FOR 19271 INSTRUCTIONS

$	0.00	CONTROL NEW	ANALYSIS $	252508.16	0	57.00
$	0.00	INPUT/OUTPUT	DESIGN	236095.13	0	47.00
$	223380.00	PRE POST CPU	CODE	273971.35	4964	45.00
$	218386.60	ALGORITHM	TEST	357299.04	5702	38.30
$	443610.00	DATA MANAGEMENT	DOCUMENT $	142667.11	7155	62.00
$	108750.00	REAL TIME			1450	75.00
$	1262540.79	TOTAL BY INSTRUCTION	TOTAL $	1262540.79		

COST $ 1262540.79 FOR 19271 INSTRUCTIONS

REFERENCES

1. Nelson and Sukert, RADC Software Data Acquisition Program, Rome Air Development Center, Griffiss AFB, NY, 20 November 1975.

2. Boehm, B. W., The High Cost of Software, "Strategies for Developing Large Software Systems," 3-14, E. Horowitz, Ed., Addison-Wesley Publishing Company, Reading, MA, 1975.

3. Dummer and Winton, An Elementary Guide to Reliability, Pergamon Press, Headington Hall, Oxford, England, 1968.

4. Newsweek. "Rising Costs of Computer Software", June 1976.

5. Information Processing/Data Automation Implications of Air Force Command and Control Requirements in the 1980's (CCIP-85), Space and Missile Systems Organization, Los Angeles, CA, (DDC Number-AD-742292), February 1972.

6. Current results from the Analysis of Cost Data for Computer Programming, Electronic System Division, Hanscom AFB, MA, DOC Number AD 637-801, August 1966.

7. Aircraft Avionics (Digital Avionics Study) Cost Trade Analysis, ASD-TR-73-18, Volume IV (Appendix C), Aeronautical Systems Division, Wright-Patterson AFB, OH, April 1973.

8. Government/Industry Software Workshop Summary Notes. Electronic Systems Division, Hanscom Air Force Base, Mass., January 1975.

9. Interim Guidance for Preparation of Cost Estimates for Tactical Software Programs, Hydrospace-Challenger, Inc., October 1974.

10. Wolverton, R. W., "The Cost of Developing Large-Scale Software," IEFE Transactions on Computers, Vol. C-23, No. 6, 615 -636, June 1974.

11. Balkovich, E. E., A Method of Estimating the Cost of Avionics Software, RM-1982, General Research Corporation, May 1975.

12. Taback and Ditmore, Estimation of Computer Requirements and Software Development Costs, RM-1873, General Research Corporation, March 1974.

13. Buck, F., et al, A Cost-By-Function Model For Avionic Computer Systems, Vol. I, NADC-SD-7088, Naval Air Development Center, March 1971.

14. Bourdon, G. A., A Proposed In-House Research Program for Improving Software Cost Estimations at ESD (Draft), Electronic System Division, Hanscom Air Force Base, Mass., April 1976.

REFERENCES (Cont)

15. Gaumer, W. F., A Preliminary Cost Analysis of the Communications Processor For the F-15 Joint Tactical Information Distribution System, *Thesis GSM/SM/76S-8, Air Force Institute of Technology*, Wright-Patterson AFB, OH: September 1976.

16. Frederic, B. C., A Provisional Model For Estimating Computer Program Development Costs, TM-7/REV 1, Tecoloti Research Inc., December 1974.

17. Malone, J. L., Estimating Software Life Cycle Costs, International Business Machines Corporation, Federal Systems Division, Westlake Village, Calif., April 1975.

18. Working Notes provided by Capt. Gerald A. Bourdon, Cost Analysis Division, Electronic Systems Division, Air Force Systems Command, Hanscom AFB, MA. Source not identified but believed attributable to Aerospace Corporation report dated April 1975.

19. Nelson, E. A., Management Handbook for the Estimation of Computer Programming Costs, System Development Corporation, DDC Number AD-648-750, March 1967.

20. Trainor, W. L., Software - From Satan to Saviour, Air Force Avionics Laboratory, Wright-Patterson AFB, Ohio, presented at the National Aerospace Electronics Conference, May 1973.

21. Getz, S., et al, A Cost-By-Function Model for Initial Navigation Systems, Vol. I, NADC-73080-50, Naval Air Development Center, April 1973.